MASTER EN SCIENCES INFORMATIQUES

Risques de sécurité liés au "Cloud Design Pattern"

Danis, Maël

Award date:
2016

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'Informatique
Année académique 2015-2016

Risques de sécurité liés au « Cloud Design Pattern »

Maël Danis

Promoteur : _____ (Signature pour approbation du dépôt – REE art. 40)

Professeur P. Thiran

Mémoire présenté en vue de l'obtention du grade de

Master en Sciences Informatiques

Résumé

Actuellement, afin d'améliorer leur rentabilité, les entreprises ont de plus en plus souvent recours au cloud pour l'élaboration de nouveaux systèmes d'information.

Cependant, comme les entreprises veulent réduire leur cout[1] au maximum, elles n'ont pas toujours conscience des risques liés à la sécurité sur le cloud et ne sont pas toujours prêtes à investir dans celle-ci. Or, ne pas prendre en compte ces risques peut mettre l'entreprise en grande difficulté tant d'un point de vue légal que financier.

Cette recherche va mettre en lumière les risques de sécurité du cloud. Nous procéderons à une analyse de différents patterns de développement liés au cloud à partir de la méthode EBIOS. Cette analyse permettra d'élaborer des contre-mesures afin d'aider les architectes et les administrateurs à développer des systèmes d'information fiables et sécurisés.

[1] La nouvelle orthographe a été utilisée pour la rédaction du mémoire.

Je tiens à remercier toutes les personnes

qui ont collaboré de près ou de loin

à l'élaboration de ce mémoire,

en apportant leur soutien et

en donnant de leur temps.

Table des matières

Table des illustrations 5
1.1 Figures 5
1.2 Tableaux 5
2 Introduction 7
3 Cloud computing 9
3.1 Présentation du « cloud computing » 9
3.2 Présentation d'un « Cloud Design Pattern » 11
3.3 Notion de sécurité d'un système d'information 13
3.4 Top des menaces liées au cloud 14
4 Méthodes d'analyse de risques informatiques disponibles 18
4.1 Méthode EBIOS 21
4.2 Méthode Méhari 22
4.3 Méthode OCTAVE 23
4.4 Méthode OCTAVE Allégro 25
4.5 Choix d'une méthode 28
4.5.1 Comparaison entre les méthodes 29
4.5.2 Choix de la méthode 30
4.6 Approfondissement de la méthode EBIOS 30
4.6.1 1er Module : la contextualisation 30
4.6.2 2ème Module : l'étude des évènements redoutés 33
4.6.3 3ème Module : étude des scénarios 33
4.6.4 4ème Module : étude des risques 34
4.6.5 5ème Module : étude des mesures de sécurité 36
5 Application d'EBIOS aux patterns du cloud 38
5.1 Etude du contexte 38
5.1.1 Cadrage de l'étude des risques 38
5.1.2 Description du contexte général 39
5.1.3 Délimitation du périmètre de l'étude 39
5.1.4 Identification des paramètres à prendre en compte 39

5.1.5 Identification des sources de menace 40
5.1.6 Identification des besoins en termes de sécurité et élaboration des échelles de besoin 40
5.1.7 Définition de l'échelle de gravité 41
5.1.8 Définition de l'échelle de vraisemblance 42
5.1.9 Etude des risques 42
5.1.10 L'identification des biens 43
5.1.11 L'identification des mesures de sécurité existantes 43
5.2 Analyse des Patterns 44
5.2.1 Méthodologie 44
5.2.2 Stateful Component 45
5.2.3 Stateless Component 49
5.2.4 User Interface Component 52
5.2.5 Elastic Load Balancer 56
5.2.6 Message Mover 59
5.2.7 Processing Component 63
5.2.8 Data Access Component 66
5.2.9 Two-Tier Cloud Application 69
5.2.10 Three-Tier Cloud Application 71
5.2.11 Hybrid Development Environment 74
6 Discussion 77
7 Conclusion 79
8 Bibliographie 81

Table des illustrations

1.1 Figures

Figure 1 : Famille Iso 27000 18
Figure 2: Cycle d'évaluation des risques 20
Figure 3: Module d'étude de risques méthode EBIOS (1- guide méthodologique, 2010) 22
Figure 4 : Méthode OCTAVE phases et processes 24
Figure 5 : OCTAVE Allegro processes [SEI, 2007] 26
Figure 6 : Stateful components (Fehling, 2014, p. 169) 45
Figure 7: Stateless components (Fehling, 2014, p. 172) 49
Figure 8 : User interface component (Fehling, 2014, p. 176) 52
Figure 9 : Elastic load balancer interacting with an elastic platform or elastic infrastructure (Fehling, 2014, p. 255) 56
Figure 10 : Message mover integrating queues of two environments (Fehling, 2014, p. 226) 59
Figure 11 : Processing component (Fehling, 2014, p. 181) 63
Figure 12 : Data access components integrating stateful components and storage offering residing in two clouds (Fehling, 2014, p. 189) 66
Figure 13 : Two-tier architecture (Fehling, 2014, p. 291) 69
Figure 14 : Three-tier architecture (Fehling, 2014, p. 295) 71
Figure 15 : Hybrid development environment and its integration with a production environment (Fehling, 2014, p. 327) 74

1.2 Tableaux

Tableau 1 : Croisement des différentes méthodes présentées avec les critères de l'ANSI [ANSI, 2014] 29
Tableau 2 : Description des critères de confidentialité. 31
Tableau 3 : Description des niveaux de gravité. 32
Tableau 4 : Description des niveaux de vraisemblance 32
Tableau 5 : Description des critères de confidentialité 41
Tableau 6 : Description des critères d'intégrité 41
Tableau 7 : Description des critères de disponibilité 41
Tableau 8 : Description des niveaux de gravité 42
Tableau 9 : Description des niveaux de vraisemblance 42
Tableau 10 : Tableau d'analyse gravité vs vraisemblance 42
Tableau 11 : Lien entre les biens essentiels et les biens supports. 43
Tableau 12 : Evènements redoutés stateful components 46

Tableau 13 : Scénario stateful components 47
Tableau 14 : Risques stateful components 48
Tableau 15 : Evènements redoutés stateless components 49
Tableau 16 : Scénarios stateless components 51
Tableau 17 : Risques stateless components 51
Tableau 18 : Evènements redoutés user interface components 53
Tableau 19 : Scénarios user interface component 55
Tableau 20 : Risques user interface component 55
Tableau 21 : Evènements redoutés elastic load balancer 57
Tableau 22 : Scénarios elastic load balancer 58
Tableau 23 : Risques elastic load banlancer 58
Tableau 24 : Evènements redoutés message mover 60
Tableau 25 : Scénarios message mover 62
Tableau 26 : Risques message mover 62
Tableau 27 : Evènements redoutés processing component 64
Tableau 28 : Scénarios processing components 65
Tableau 29 : Risques processing components 65
Tableau 30 : Evènements redoutés data access components 67
Tableau 31 : Scénarios data access components 68
Tableau 32 : Risques data access components 68
Tableau 33 : Evènements redoutés two-tier architecture 69
Tableau 34 : Scénarios two-tier architecture 70
Tableau 35 : Risques two-tier architecture 70
Tableau 36 : Evènements redoutés three-tier architecture 72
Tableau 37 : Scénarios three-tier architecture 73
Tableau 38 : Risques three-tier architecture 73
Tableau 39 : Evènements redoutés hybrid development environment 75
Tableau 40 : Scénarios hybrid development environment 76
Tableau 41 : Risques hybrid development environment 76

2 Introduction

A l'heure où la rentabilité économique est devenue la priorité des entreprises, ces dernières ont de plus en plus souvent recours au cloud pour l'élaboration de nouveaux systèmes d'information.

Derrière le mot « cloud » se cache l'idée d'utility computing mis en avant par IBM [Rappa, 2004]. L'utility computing conçoit l'informatique comme un service au client devant répondre à un certain nombre de critères : nécessité, disponibilité, facilité d'utilisation. De plus, le fondement de l'utility computing est lié à son taux d'utilisation, à son évolutivité et à une exclusivité du service, ce qui permet de réduire les couts de l'entreprise.

Le cloud est donc un service informatique disponible via internet ; nous parlons de « cloud service provider ». Il s'agit en fait d'entreprises vendant ce service. Le cloud service provider devient donc un partenaire important pour la viabilité d'une entreprise qui utilise le cloud.

Il est utilisé par les entreprises pour déployer leur système d'information. En effet, c'est l'avenir de l'hébergement d'infrastructures et applicatifs car le cloud permet à la fois un gain financier et une flexibilité accrue.

Cependant, les entreprises, voulant réduire leur cout au maximum, n'ont pas toujours conscience des risques liés à la sécurité sur le cloud et ne sont pas toujours prêtes à investir dans celle-ci. Or, ne pas prendre en compte ces risques peut mettre l'entreprise en grande difficulté tant d'un point de vue légal que financier.

Les études déjà réalisées sur les risques liés à la sécurité du cloud se sont attachées à l'infrastructure alors que cette recherche s'intéresse aux patterns architecturaux. Elle tente de mettre en lumière les risques liés à la sécurité à partir d'une analyse de différents patterns architecturaux. Ceux-ci sont des solutions génériques répondant à un certain nombre de problèmes liés aux propriétés du cloud.

Un risque est composé de trois éléments : une vulnérabilité, une source de menace ainsi qu'une probabilité. Ces trois composantes réunies désignent le niveau de risque qui pèse sur un bien. Afin de déterminer le risque lié à un bien, il existe différentes méthodes telles que OCTAVE, EBIOS, MEHARI, … Ces méthodes sont des guides méthodologiques à destination des responsables en sécurité de systèmes d'information. Leur but est de procéder à une étude des biens qui sont sous leur responsabilité afin d'adapter au mieux les stratégies de sécurisation.

Dans le cadre de ce mémoire, nous avons procédé à une analyse de différents patterns de développement liés au cloud à partir de la méthode EBIOS. Les trois patterns architecturaux

retenus dans cette étude ont été expérimentés en entreprise. Il s'agit des patterns suivants : « two-tier cloud application », « three-tier cloud application » et « hybrid development environment ». Chacun de ces trois patterns est également composé de différents patterns : «stateful component », « stateless component », « user interface component », « elastic load balancer », « message mover », « processing component » et « data access component ». Comme ces derniers ne sont pas systématiquement repris dans chaque pattern architectural, ils ont été analysés de manière indépendante.

Pour l'analyse des patterns, la méthodologie utilisée comprend six points :

- présenter le pattern ;
- lister les évènements redoutés ;
- évaluer les évènements redoutés ;
- créer des scénarios pour chaque évènement ;
- évaluer ces scénarios ;
- analyser les risques.

Dans un premier temps, il s'agit de présenter le pattern en l'accompagnant d'un schéma issu du livre « Cloud Computing Patterns » [Fehling, 2014]. Ensuite, les évènements redoutés applicables ont été listés. Soit ceux-ci ont été expérimentés au sein de deux grandes entreprises à travers ma fonction d'administrateur middleware, soit ils sont issus de la littérature scientifique. Ensuite, ces évènements redoutés ont été évalués. Ils ont reçu une note de gravité alors que les scénarios qui ont été créés pour chaque évènement ont obtenu une note de probabilité lors de leur évaluation. Afin d'aider à leur attribution, chaque note a été définie au préalable à partir des définitions reprises dans la méthode EBIOS. Enfin, ces deux notes ont permis d'établir un niveau de risque pour chaque scénario. Ce niveau permet de guider les choix en matière de patterns mais aussi en termes de mesures de sécurité à prendre dans le cas où le pattern architectural en question est sélectionné par l'architecte.

Par la suite, cette analyse permettra d'élaborer des contre-mesures afin d'aider les architectes et les administrateurs à développer des systèmes d'information fiables et sécurisés.

3 Cloud computing

Dans un premier temps, ce chapitre présentera le cloud et ses propriétés. Dans un second temps, nous définirons un « Cloud design pattern » et nous présenterons un exemple. Ensuite, nous poursuivrons par une introduction sur la sécurité au sein des systèmes d'information. Enfin, nous réaliserons une synthèse des menaces de sécurité qui courent actuellement sur le Cloud et nous terminerons par les différentes problématiques de sécurité du cloud.

3.1 Présentation du « cloud computing »

Ces dernières années, le cloud computing est devenu un buzzword. Derrière ce mot se cache l'idée d'utility computing mis en avant par IBM [Rappa, 2004]. L'utilty computing voit l'informatique comme un service au client qui devra donc répondre à un certain nombre de critères : nécessité, disponibilité, facilité d'utilisation. L'utilty computing tient également compte du taux d'utilisation, de l'évolutivité et d'une exclusivité du service.

Cette vision tient du modèle économique où le client payera le service en fonction de sa consommation. Il ne payera que lorsqu'il utilise le service. La mutualisation de l'infrastructure permettra au client de bénéficier des économies d'échelle.

La propriété d'exclusivité de service fera en sorte que peu de providers seront capables de produire un service concurrent, car le prix du service dépendra fortement de la taille de l'infrastructure. Plus un datacenter est grand, plus le cout d'un cycle CPU[2] et le prix de l'espace de stockage diminuent [Armbrust, 2009]. Nous nommerons donc « service provider » toutes entreprises mettant à la disposition de ses clients une infrastructure de type Cloud.

L'idée du cloud est de proposer un service informatique à la demande qui est disponible la plupart du temps par le biais d'internet. Ces services peuvent être délivrés sous différentes formes [NIST, 2011] :

- « Infrastructure as a Service » (IaaS) : cette offre de service fournira un service jusqu'au système d'exploitation ; typiquement, on commandera une machine virtuelle ;
- « Platform as a Service » (PaaS) : cette offre de service gérera la partie middleware sur laquelle les applications des différents clients pourront être déployées. (ex : openshift, google app Engine,…) ;

[2] Central processing unit

- « Software as a Service » (SaaS) : le service disponible sera une application complète. C'est la forme la plus connue du grand public (ex : gmail, office online, …).

Afin d'aider l'utilisateur de ces différents services à être autonome, le cloud service provider offrira un certain nombre de caractéristiques communes [Fehling, 2014].

- **« On demande self-service » :** l'utilisateur sera capable à l'aide d'API[3] de gérer son propre service. La notion de provisions et de décommissions de ressources par l'utilisateur est primordiale ; le but étant de gérer au mieux les ressources allouées en fonction de la charge demandée au service.
- **« Broad network access » :** afin de garantir les temps d'accès au service proposé, le système cloud n'est pas dépendant de la localisation de l'utilisateur. On rapprochera au plus près les infrastructures et leurs utilisateurs. Inutile de stocker les données européennes dans un data center de l'autre côté de l'Atlantique et les données américaines en Europe, cela générerait un trafic inutile.
- **« Measured service (Pay per user) » :** le cloud computing est un nouveau modèle économique qui est basé sur l'utilisation réelle et non sur la propriété. Dans cette optique, il sera donc très important pour un CSP[4] de mesurer la consommation réelle de chaque client. Cette caractéristique est fortement liée au fait que le client peut provisionner et décommissionner lui-même des ressources. Cela permet de payer uniquement les ressources utilisées.
- **« Resources Pooling » :** dans le but d'offrir un service à la demande, il est important que l'infrastructure sur laquelle repose le service soit capable d'assigner ces ressources de façon automatique. Pour ce faire, les ressources physiques seront partagées entre les différents clients. La notion de multi-tenant sera importante afin de garantir le cloisonnement entre les clients. Ce point sera crucial dans l'élaboration de la sécurité du service proposé.
- **« Rapid elasticity » :** l'idée du cloud étant de faire des économies d'échelle via les « Resources Pooling », il est important que chaque client puisse augmenter ses ressources au moment où il en a besoin. Ainsi, quand un client a besoin de moins de ressources, il les laisse à disposition des autres utilisateurs. On aura ainsi une élasticité des ressources sur le plan horizontal et non vertical.

Le fait de tourner sur un environnement évolutif va avoir un impact sur l'architecture de l'application, qui devra respecter un certain nombre de propriétés en lien direct avec les propriétés du cloud [Fehling, 2014].

[3] Application programing interface
[4] Cloud Service Provider

- **« Distribution » :** de par la nature du cloud qui peut être évolutif horizontalement, la propriété de distribution est très importante. Ces différents composants seront distribués sur les ressources allouées à l'application.
- **« Isolated States » :** propriété en lien direct avec la propriété précédente car pour pouvoir permettre l'élasticité et donc l'apparition ou la disparition d'une instance de programme à tout moment, l'application doit être stateless afin que cette perte de l'instance ne génère pas d'erreurs lors de la perte d'une session.
- **« Automated management » :** étant donné la propriété d'élasticité du cloud, l'ajout et la suppression d'instances d'application sont communes. Il faut donc prendre en considération l'automatisation de cette tâche, et ce grâce à l'utilisation d'un monitoring sur les instances qui rend efficace l'utilisation des ressources sur lesquelles le cloud est basé.
- **« Loose coupling » :** le but est de minimiser au maximum les dépendances inter applications afin de garantir que l'application ne soit pas en erreur lorsque l'instance d'un composant n'est plus disponible de par l'élasticité, propriété déjà connue dans le cadre de l'utilisation de web service et du style SOA[5].

Pour répondre à ces différentes propriétés, différents ouvrages sont consacrés au cloud design pattern. Un pattern étant une solution générique pour un problème posé dans un contexte particulier [Fehling, 2014], [Homer, 2014]. Toutes ces propriétés amènent à utiliser des patterns architecturaux afin de garantir le bon fonctionnement des applications déployées sur le cloud.

Le cloud est un service décentralisé, ce qui pose des questions quant au niveau de sécurité qui devra être fourni. Cette problématique motive une analyse de risques appliquée à différents « Cloud Computing Patterns », afin d'extraire les risques encourus lors de l'utilisation de chaque pattern.

3.2 Présentation d'un « Cloud Design Pattern »

Un design pattern peut être vu comme une solution générique à destination des architectes. Ceux-ci vont y trouver des solutions existantes qui vont leur servir de base pour construire une solution complète pour un problème donné [Fehling, 2014]. Le concept de design pattern est un outil qui s'applique à différentes souches du génie logiciel (architecture, application, sécurité, …). La composition d'un design pattern est très structurée. Elle contiendra toujours les éléments suivants :

- **« Name » :** nom du pattern ;
- **« Driving question » :** question qui pose le problème lié au pattern ;

[5] Service Oriented Architecture

- **« Context » :** contextualisation d'une situation problématique que le pattern va résoudre ;
- **« Solution » :** brève description de la façon dont le design pattern va résoudre le problème posé dans la contextualisation. Cette description permet à l'utilisateur de comprendre, à l'aide du schéma d'architecture, comment fonctionne le pattern;
- **« Result » :** solution complète qui présente les détails de l'implémentation, elle peut aussi être complétée d'un exemple concret (un morceau du code) ;
- **« Variation » :** petite variation d'implémentation mais qui ne justifie pas un nouveau pattern ;
- **« Related patterns » :** référence vers d'autres patterns qui peuvent compléter la solution en apportant de nouvelles propriétés aux patterns, par exemple « loose-coupling », « security », … ;
- **« More information » :** exemples de produits ou de services déjà disponibles et utilisant la solution.

Les design patterns ainsi présentés sont donc assemblés entre eux afin de produire une solution complète rencontrant tous les besoins préalablement définis. C'est le rôle de l'architecte de l'application d'assembler une solution.

Voici un exemple issu du livre « Cloud Design Pattern » [Fehling, 2014].

- **« Name »** : Management Configuration.
- **« Driving question »** : Comment garder les configurations entre les différentes instances applicatives ?
- **« Context »** : dans le cadre d'une infrastructure élastique, il est important de pouvoir propager un changement sur toutes les instances et aussi garantir que celle-ci ait les mêmes paramètres tout le temps.
- **« Solution » :** la solution est de mettre en place une base de données avec tous les paramètres relatifs à chaque application ; deux solutions sont possibles pour propager ces informations aux instances, soit l'instance va chercher elle-même dans la base de données ses paramètres, soit c'est un agent qui propage les configurations au sein des instances.
- **« Result » :** le choix d'une des deux solutions va être déterminé en fonction du nombre de fois où des changements vont être effectués dans la configuration. Dans le cas où un des changements doit être propagé immédiatement, il préférable d'utiliser la méthode push. Par contre, lors des changements qui ne sont pas urgents, on utilisera la méthode pull car elle réduit la charge sur l'infrastructure.
- **« Related Patterns »: «** Blob starage **»**, **«** relational database **»**, **«** key-value storage **»**.
- **« Information » :** outils disponibles : « Puppet », « Chef », « Ansible ».

3.3 Notion de sécurité d'un système d'information

La sécurité au sein d'un système d'information consiste à garantir un certain nombre de propriétés sur le système et l'information qui le composent [Collin, 2015]. Afin d'y arriver, différentes actions devront être prises en compte durant tout le cycle de vie du système d'information, et ce du début de l'analyse jusqu'à la fin de vie du système.

Les enjeux sont donc importants car, dans la plupart des cas, une indisponibilité grave ou une perte importante de données conduit à la fermeture de la société impliquée sans oublier que le responsable du SI[6] pourrait être poursuivi d'un point de vue légal.

Voici les principaux critères de sécurité liés à l'information qui doivent être pris en compte lors d'une analyse de sécurité :

- **la confidentialité :** garantie que les informations stockées dans le système et transitant entre les composants seront disponibles et consultables uniquement par les personnes autorisées ;
- **l'authenticité :** propriété qui valide l'identité du prétendant et qui est très complémentaire avec la confidentialité car ce n'est qu'après l'authentification d'une entité que la confidentialité peut être garantie ;
- **l'intégrité :** l'intégrité des données valide le fait qu'elles sont exactes et qu'elles n'ont pas été modifiées à posteriori ;
- **la disponibilité :** le système d'information doit pouvoir assurer le service pour lequel il a été conçu pour tous les utilisateurs autorisés ;
- **la traçabilité :** garantir la traçabilité de tous les accès aux données. Souvent liées à des contraintes légales, ces traces (logs) seront utilisées dans le cadre d'enquêtes.

Dans le cadre de la sécurisation d'un système d'information, ces différentes propriétés sont nos objectifs. Le processus de sécurisation est composé de trois parties.

- **La prévention :** première étape dans la sécurisation d'un système d'information qui va commencer en même temps que la construction du système lui-même ; le but étant de construire un système sans vulnérabilité répondant aux propriétés citées précédemment.
- **La détection :** cette étape suit la mise en production. Elle a pour but de détecter toutes les anomalies en fonction des différents critères de sécurité choisis. Une série de mesures va donc être mise en place dans ce but. Pour chaque critère de sécurité, il sera important de voir comment on peut détecter s'il est bien respecté ou non. Le monitoring du système tout au long de son cycle de vie est donc primordial.

[6] Système d'Information

- **La correction :** mesure pour corriger les anomalies détectées sur la base de la surveillance. On est donc dans une phase réactive.

3.4 Top des menaces liées au cloud

Dans le cadre du développement du cloud, la CSA[7] a publié un document reprenant les différentes menaces qui pèsent sur le cloud [CSA, 2013]. Les menaces sont ainsi classées par ordre de priorité.

- **« Data Breaches »**

  Dans la grande majorité des systèmes d'information, les données forment le cœur du système. Dans la plupart des cas, ces données peuvent être sensibles pour différentes raisons. Dans le cadre de l'utilisation du cloud, l'accès à ces données par des personnes mal intentionnées peut être un risque.

- **« Data Loss »**

  Il s'agit d'une perte de données ou d'une perte d'accès aux données. A partir du moment où les données sont stockées dans le cloud, leur sauvegarde est dépendante du « service provider ». Un incident peut provoquer la perte de ces données. Un autre souci peut survenir lorsque les données sont chiffrées avec une clé. Si cette clé est perdue, les données le seront également. Dans ce cas, cette clé est sous la responsabilité du client, et ce malgré un backup mis en place sur le cloud.

- **« Account or service traffic hijacking »**

  Cette menace plane sur l'utilisateur dès qu'un service demande une authentification, ce qui est le cas du cloud. En plus, les tentatives d'accès frauduleux sont d'autant plus fortes qu'il n'y a pas d'accès physique. L'authentification forte est d'ailleurs recommandée pour ce type de service.

[7] Cloud Security Alliance

- **« Insecure interfaces and APIs »**

Les « cloud service provider » ont pour habitude de fournir une interface et/ou Api à leurs clients dans le but de leur permettre de gérer leurs ressources. On trouve généralement la possibilité de gérer le provisioning, le monitoring, voire même les accès. Cela explique que cette interface avec le monde extérieur est un autre point d'entrée pouvant faire l'objet d'attaques.

- **« Denial of Service »**

L'attaque « Denial of service » a pour but de rendre inaccessible un site web ou un service. La principale technique consiste en une saturation des ressources. Dans ce cas, l'utilisateur peut ne plus avoir accès à ses données ou encore au service dont il est dépendant pour son business, ce qui entraine un arrêt de production. Un autre impact possible peut être l'explosion de la facture. En effet, dans le cloud, le paiement dépend de la consommation des ressources. Cela peut entrainer des factures impayables pour les petites structures. Ces dernières années, l'utilisation de ce type d'attaques de façon distribuée (DDOS)[8] rend la détection complexe et augmente aussi la puissance de ces attaques [ARBOR, 2015].

- **« Malicious insiders"**

Le CERT définit cette menace comme suit: "*A malicious insider threat to an organization is current or former employee, contractor, or other business partner who has or had authorized access to an organization's network, system, or data and intentionally exceeded or misused that access in a manner that negatively affected the confidentiality, integrity, or availability of the organization's information or information systems."* [CERT, 2016].

La définition donnée par le CERT montre bien que le facteur humain est une source de faille. Dans le cloud, ce facteur est exacerbé pour la simple raison que le service est délocalisé. De ce fait, il peut être difficile de garantir l'intégrité du personnel.

- **« Abuse of cloud services »**

Dans ce cas, ce n'est pas le client du cloud service provider qui peut être la cible, mais bien le cloud service provider même. Celui-ci est utilisé à des fins illicites notamment lorsqu'on utilise la puissance de calcul pour le cassage de clé ou la propagation de malware ou encore pour mener des attaques de type DDOS. Le cloud service provider

[8] Distributed Denial Of Service

doit donc mettre en place un système de détection d'abus et définir ce qu'est un abus. Les contrats devront être validés par un service juridique afin d'éviter toute ambiguïté et de répondre au besoin réel de l'entreprise.

- **« Insufficient Due Diligence»**

Le manque de connaissance est un facteur clé dans l'ouverture de failles. L'utilisation du cloud promet aux entreprises une réduction de leurs couts. Dès lors, elles peuvent être tentées de faire migrer l'ensemble de leur infrastructure applicative sur le cloud. Cependant, c'est à ce moment que l'on peut ouvrir des brèches. En effet, à l'heure actuelle, la majorité des architectes, des développeurs et des administrateurs travaillant dans ce domaine ne sont ni formés aux contraintes, ni familiarisés avec celles que le cloud implique. En conséquence, les entreprises qui adoptent l'utilisation de service cloud doivent disposer des ressources nécessaires leur permettant de comprendre les tenants et aboutissants du cloud et ainsi être en mesure de réduire les risques.

- **« Shared Technology Vulnerabilities »**

En fonction du modèle de cloud (IaaS, PaaS et SaaS), les différents clients du service provider vont partager les couches inférieures. Une mauvaise configuration ou une faille sur le software se cachant derrière peut entrainer une faille de sécurité pour tous les clients, surtout si le client n'est pas au courant des technologies cachées derrière (black box). Il faut donc que le service provider surveille en permanence la sécurité de son offre. Par exemple, une faille au niveau de l'hyperviseur se répercute sur tous les services déployés au-dessus [Perez-Boteron, 2013].

Ce classement représente les différentes menaces possibles. Malheureusement, cette liste n'est pas suffisante pour permettre une migration vers le cloud. Il s'agit plus d'une mise en lumière des différentes problématiques et des dérives qui peuvent découler d'une mauvaise compréhension des caractéristiques du cloud.

Voici quelques exemples qui montrent que ces failles sont présentes à tous les niveaux de l'architecture.

Dans le cadre d'un « malicious insider », il est simple pour une personne ayant accès au réseau et aux machines de pouvoir détourner de l'information ou de mettre en place des systèmes pour y arriver. Minh-Duong Nguyen (2014) met en avant trois types d'attaques possibles pour une personne ayant déjà accès au système [Nguyen, 2014]. Il explique la démarche à appliquer dans le cadre de ces attaques : « *memory dump scanning »* permettant de retrouver les mots de passe en clair au sein de la mémoire, « *Public Template*

Poisoning » dont le but est de modifier l'image qui sera utilisée par l'utilisateur du cloud afin qu'il puisse démarrer un programme malveillant. Enfin, le dernier scénario proposé met en avant le « *snapshot cracking »* qui consiste à accéder aux « snapshot » sorte de backup des clients. Cela est très facile en copiant l'image de la machine virtuelle du client et en la démarrant à l'aide d'outils, ce qui permet d'avoir un accès complet à l'information qu'elle contient.

Dans un autre article [Kholia, 2013], il est montré que la sécurité d'un service cloud peut être mise à mal par le client (logiciel) qui en général est fourni par le « cloud service provider ». Dans le cadre du service de stockage *Dropbox,* cet article a mis en cause la sécurité de différentes versions du client. Ce client (logiciel), étant disponible au public, peut facilement subir un « revers engineering » pouvant être exploité dans le but de trouver des failles dans le protocole d'échange avec le service « *backend »*. Le code décompilé aura été utilisé afin d' « hijacking » le compte d'un utilisateur.

Suite à la décompilation du code du client, l'authentification à deux facteurs n'est pas présente au sein du client bien qu'elle le soit sur le site internet de Dropbox. Le client se contente d'un ID d'utilisateur, ce qui permet de se connecter sans difficulté à la place d'un autre utilisateur. Il suffira de connaitre le numéro lié à chaque utilisateur.

De plus, le nombre d'attaques DDOS est de plus en plus courant, et ce avec une intensité de plus en plus importante. « *Again this year, the proportion of respondents seeing attacks targeting cloud-based services has grown, up from 19 percent two years ago, to 29 percent last year and now 33 percent this year — a clear trend.* » [ARBOR, 2015]. En effet, il est facile de commander une attaque DDOS sur internet car ce type d'attaque est disponible sur le marché noir. Ou encore, ce cas expérimenté en entreprise où il a été démontré qu'une attaque « slow loris » est très simple à mettre en place via une simple ligne ADSL pour peu que le service internet n'y soit pas préparé. Cette expérimentation a été réalisée afin de valider une contre-mesure. Ces deux attaques n'entrainent qu'une indisponibilité du service. En 2015 et 2016, les sites du gouvernement belge ont subi ce type d'attaques. Elles ont été revendiquées par « downsec ».

Ces différents exemples d'exploitation de la vulnérabilité de différents services disponibles sur internet rendent légitime la volonté de garantir un service cloud de qualité et sécurisé afin d'offrir un service répondant aux différents critères de sécurité, ce qui nous amène donc à l'analyse des risques.

4 Méthodes d'analyse de risques informatiques disponibles

Après avoir évoqué brièvement en quoi consistait la sécurité au sein d'un système d'information, ainsi que les vulnérabilités engendrées par l'utilisation du cloud, nous pouvons conclure que les besoins en termes de sécurité sont bien présents.

Afin de mieux évaluer les risques encourus par l'utilisation du cloud dans le cadre du développement d'applications, ce chapitre va être consacré à l'évaluation des risques au sein d'un système d'information.

Pour ce faire, nous introduirons le risque par une présentation succincte de la famille de la norme ISO 27000, celle-ci étant à la base de toutes les méthodes d'analyse de risques. Puis, nous terminerons par la présentation de différentes normes possibles pour évaluer un système d'information.

Figure 1 : Famille Iso 27000[9]

La famille des normes ISO 27000 est composée comme suit :

- **ISO 27000 :** document présentant la norme et comprenant aussi le glossaire ;
- **ISO 27001 :** document contenant la norme de sécurisation d'un système d'information ;
- **ISO 27002 :** guide de bonnes pratiques ;

[9] (http://www.neupart.com/hubfs/Images_/iso27001.jpg?t=1467114823900)

- **ISO 27003 :** guide permettant de mettre en place la norme sur son système ;
- **ISO 27004 :** guide contenant les informations nécessaires afin de mesurer le niveau de sécurité du système ;
- **ISO 27005 :** guide pour la gestion des risques ;
- **ISO 27006 :** expression des besoins afin d'obtenir une certification.

Cette standardisation a pour but d'uniformiser la notion de risques en aidant à les identifier. Une partie de la spécification permettra d'émettre des recommandations afin de pouvoir passer une certification sur son système d'information. Il y a donc plusieurs parties (voir le schéma ci-dessus).

Afin d'identifier un risque, nous allons le définir de manière la plus précise possible.

« Il y a risque de sécurité de l'information lorsque l'on a conjointement :

- une source de menace ;
- une menace ;
- une vulnérabilité ;
- un impact.

On peut ainsi comprendre qu'il n'y a plus de risque si l'un des facteurs manque » [SGDN / ANSSI / ACE / BAC, 2010].
L'importance d'un risque est liée par le niveau de criticité de chaque composante. Il y a donc un lien de causalité entre eux.

Il existe deux catégories de facteurs. La première peut être considérée comme une cause. Dans cette catégorie, nous retrouverons les éléments suivants : une source de menace, une menace et une vulnérabilité. La seconde peut être considérée comme une conséquence qui sera donc l'impact.

Afin de diminuer le risque, nous devons travailler selon plusieurs angles : soit réduire les sources de problèmes en limitant la menace, soit réduire les vulnérabilités en travaillant sur les conséquences qu'une attaque pourrait produire par l'adoption de mesures [Collin, 2015].

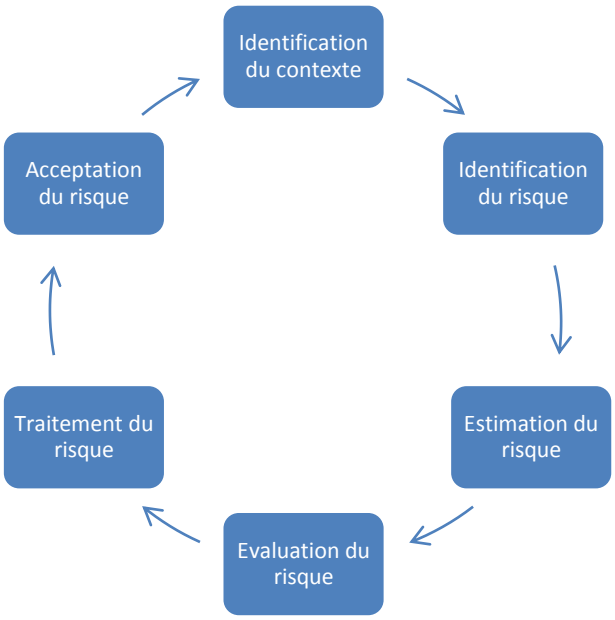

Figure 2: Cycle d'évaluation des risques

Une analyse de risques commence toujours par l'identification du contexte [Mayer, 2006]. C'est à cette étape que l'on va définir l'analyse, et ce sous plusieurs angles. On commence par l'élaboration d'un cadre d'étude qui définit le contexte, les attentes (rapport, mise en avant de solutions, guide line, …) et une structure de travail. Une délimitation de l'analyse est définie afin de garantir que les objectifs de sécurité soient bien atteints tout en restant concentré sur un périmètre établi.

L'étape d'identification des risques fait appel à la définition d'un risque qui est la résultante de plusieurs facteurs (menace, vulnérabilité, impact). Il faut donc identifier l'importance des actifs au sein du contexte, ceux-ci étant considérés comme vulnérables. Une fois que les vulnérabilités sont mises en avant, on peut voir par quoi elles peuvent être menacées. Nous avons ainsi identifié la menace. Une fois que les actifs vulnérables et les menaces auxquels ils doivent faire face sont listés, nous pouvons évaluer l'impact au cas où cette menace deviendrait réelle. L'identification des risques est ainsi faite.

Lors de l'étape suivante, il faut estimer l'importance des risques à partir de deux critères : la probabilité qu'une menace soit réalisée et la gravité de l'impact. Nous pouvons alors placer nos risques au sein d'un graphe à deux dimensions qui met en évidence les risques sérieux.

L'évaluation des risques consiste en un classement de ceux-ci afin de pouvoir les traiter de façon cartésienne. Il faut aussi définir les actions à mettre en œuvre pour annuler le risque ou tout au moins le réduire au maximum. Ceci se fait en corrélation avec les budgets alloués ainsi qu'avec les priorités de l'entreprise.

Le traitement des risques consiste en l'application des différentes corrections élaborées. Dans certains cas, on peut tout simplement accepter le risque tel qu'il est ou encore le déléguer via un contrat d'assurances.

Cette analyse est cyclique, elle doit être suivie et rééditée car les trois facteurs d'un risque ne sont pas statiques.

La gestion d'un risque a pour but d'avoir une approche stratégique et cartésienne de la problématique de risque. Cela facilite la prise de décision en termes de priorisation.

La norme ISO 27000 n'est pas une méthode en soi, mais une série de lignes de conduite qui permet de pouvoir faire une analyse de gestion des risques sur un système d'information. La suite de ce chapitre sera consacrée à la présentation de quelques méthodes d'analyse présentes sur le marché. Celles-ci apportent une aide pour réaliser l'analyse.

Voici quelques méthodes d'analyse de risques au sein d'un système d'information. Cette liste n'est pas exhaustive. En effet, il en existe plus de 200 à travers le monde [Mayer, 2006] [ANSI, 2014]. Certaines de ces méthodes ne sont plus suivies ou ne sont pas connues.

- EBIOS
- MEHARI
- OCTAVE
- OCTAVE ALLEGRO
- …

4.1 Méthode EBIOS

La méthode EBIOS (Expression des Besoins et Identification des Objectifs de Sécurité) est une propriété du secrétariat général de la défense et de la sécurité nationale français. La première publication du guide date du 02/1997, une deuxième version a suivi en 2004, celle-ci convergeait vers la norme ISO 154408. Quant à la dernière version, celle qui nous intéresse, elle a été éditée le 26/01/2010 et converge vers les normes ISO 27001, ISO 27005, ISO Guide 73 et ISO 31000 [SGDN / ANSSI / ACE / BAC, 2010]. En fait, la présentation de la méthode a été revue en profondeur et les différents modules ont été adaptés.

L'objectif de la méthode est de fournir une base quant à la gestion des risques, et ce avec une certaine méthodologie, tout en répondant à la norme ISO 27001.

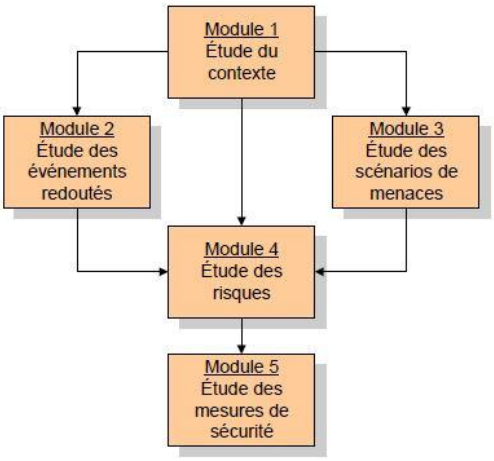

Figure 3: Module d'étude de risques méthode EBIOS (1- guide méthodologique, 2010)

La méthode est donc composée de **cinq modules**.

1. **Etude du contexte** : module qui définit le contexte dans lequel la méthode va être appliquée.
2. **Etude des évènements redoutés** : sorte de listing des menaces plausibles.
3. **Etude des scénarios de menaces** : inventaire de scénarios par menace.
4. **Etude des risques** : listing et évaluation des risques sur base des données récoltées dans les deux précédents modules.
5. **Etude des mesures de sécurité** : mise en place de mesures sur base du listing des risques établis à l'étape précédente.

Cette méthode sera décrite plus en détail dans le corps du mémoire car elle sera utilisée dans le cadre de l'analyse. La justification de ce choix sera explicitée au point « 4.7 Choix de la méthode ».

4.2 Méthode Méhari

Cette méthode a été mise au point par le CLUSIF[10]. Elle a été construite dans le but de sécuriser un SI[11] [CLUSIF, 2010].

Méhari est donc une méthode complète validant toute la sécurité au sein d'une organisation. Il s'agit d'une approche de type dichotomique (oui/non) afin de pouvoir comparer plusieurs occurrences d'audit. Elle est articulée autour de quatre axes : l'objectif, les résultats attendus, la description des mécanismes et des solutions, et enfin la qualité de service.

[10] Club de la Sécurité de l'Information Française
[11] Système d'Information

La méthode est découpée en quatorze domaines. Pour chacun d'eux, des objectifs sont définis avec un résultat attendu. Pour chaque objectif, tous les services impliqués sont déterminés afin d'atteindre le résultat voulu.

Une fois que tous les services sur lesquels une analyse doit être réalisée sont identifiés, il faut travailler au plan de sécurisation des services concernés en rappelant l'objectif pour chaque service. Quant aux résultats attendus, ils sont plus détaillés, ce qui permet de passer à la troisième étape : la description des mécanismes et solutions.

Cette étape procède à la description de toutes les mesures à appliquer afin d'atteindre le résultat voulu. Deux catégories de mesures sont dégagées: les organisationnelles et les techniques. Chaque mesure est détaillée afin de garantir une bonne compréhension.

La dernière étape consiste en l'élaboration des critères de qualité. Ils sont au nombre de trois : l'efficacité, la robustesse et la mise sous contrôle.

4.3 Méthode OCTAVE

OCTAVE[12] est une méthode [SEI, 2003] d'évaluation des risques de sécurité au sein d'organisations. Elle a été développée par «Software Engineering Institute ». Il existe une variante appelée OCTAVE-S qui est plus adaptée aux petites structures. Cette méthode fournit un questionnaire permettant de déterminer quelle méthode, OCTAVE ou OCTAVE-S est la plus adaptée à telle entreprise.

[12] Operationally Critical Thread, Asset and Vulnerabilty Evaluation

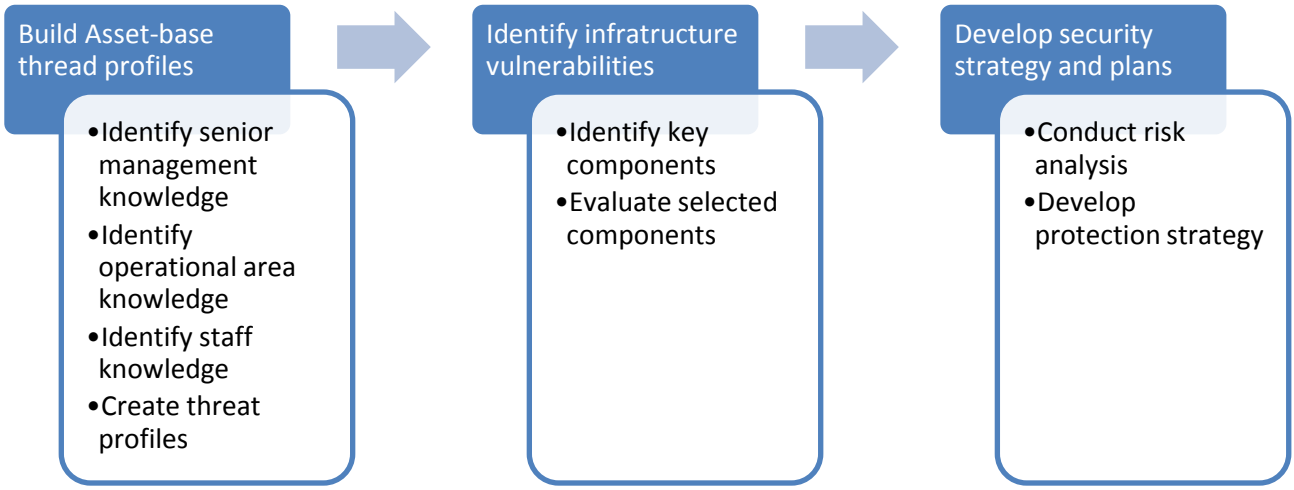

Figure 4 : Méthode OCTAVE phases et processes

Cette méthode est composée de trois phases et de huit processes.

La première phase « Build Asset-based thread profile » consiste à récolter les informations sur les biens de l'entreprise et d'établir les vulnérabilités qui y sont liées. Ces informations sont collectées auprès de différents groupes de personnes.

- **Process 1** : Identify Senior Management Knowledge – collecte d'informations sur les actifs importants ainsi que sur les besoins en terme de sécurité auprès des différents seniors managers ;
- **Process 2** : Identify Operational Area Knowledge – la collecte d'informations dans ce cas sera effectuée auprès des managers des différentes entités opérationnelles ;
- **Process 3** : Identify Staff Knowledge – la collecte d'informations relatives aux biens importants et aux vulnérabilités se fait auprès des membres des équipes concernées.
- **Process 4** : Create threat profile - une fois toutes les informations récoltées auprès des différents groupes de travail, l'équipe d'analystes sélectionne les biens les plus critiques et établit un profil de vulnérabilité pour chaque bien.

La seconde phase « Identify infrastructure vulnérabilities » est consacrée à l'analyse des différents composants supportant les éléments critiques et qui ont été mis en avant lors de la première phase. Cette deuxième phase se focalise plus particulièrement sur les aspects techniques et technologiques des composants identifiés.

- **Process 5** : Identify key components – la phase débute avec l'identification des différents composants qui supportent les biens critiques évoqués lors de la première phase ;
- **Process 6** : Evaluate selected components – il s'agit de réaliser un audit des composants et d'analyser les résultats obtenus afin de réévaluer les profils de vulnérabilité.

La troisième phase « Develop security startegy and plan » consiste en une évaluation des différents risques répertoriés et d'en sortir un plan d'actions afin de protéger efficacement le système.

- **Process 7** : Conduct risk analysis – il s'agit d'étudier les risques pour chaque bien en fonction des analyses réalisées préalablement : évaluation des vulnérabilités et impact qu'elles pourraient avoir ;
- **Process 8** : Develop protection strategy – sur la base de l'analyse des différents risques, un plan d'actions est établi afin de sécuriser les actifs de la société. Il est ainsi possible de cibler le composant faible ou ayant besoin de plus de sécurité.

4.4 Méthode OCTAVE Allégro

OCTAVE Allegro [SEI, 2007] est une nouvelle version d'OCTAVE présenté précédemment. Cette nouvelle version a pour but d'optimiser la méthode afin de diminuer les couts en temps et en ressources lors de la réalisation d'un audit. En comparaison avec l'ancienne version, l'accessibilité de la méthode a évolué afin de réduire le temps d'apprentissage.

OCTAVE Allegro est composé de huit étapes axées sur quatre domaines d'activités.

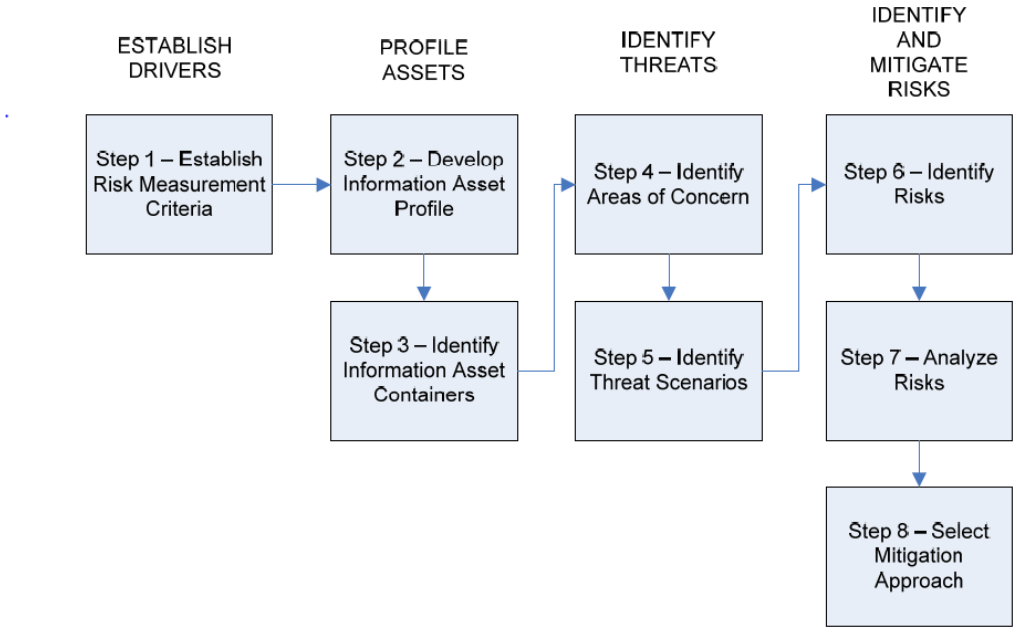

Figure 5 : OCTAVE Allegro processes [SEI, 2007]

- **Étape 1 : "Establish Risk Measurement criteria"**

Dans cette première étape, la méthode OCTAVE Allegro permet d'établir un certain nombre de critères susceptibles de faire courir un risque à l'organisation. Ces derniers sont analysés.

La méthode fourni un template afin de guider l'utilisateur dans sa démarche. Après avoir identifié les critères, il faut les classer en fonction de leur impact potentiel. (Exemples de critères extraits de la méthode : finance, réputation, productivité …)

- **Étape 2 : « Develop information asset profile »**

Dans cette étape, un listing exhaustif des biens de la société est dressé. Un bien est un actif utilisé par l'entreprise dans le but d'atteindre ses objectifs. L'exhaustivité de la liste des actifs dépend du niveau où l'analyse est effectuée dans l'entreprise. Une fois cette liste obtenue, il faut classer ses actifs en fonction de leur criticité pour assurer la réussite de l'entreprise.

Pour chaque bien, une fiche descriptive est établie (Worksheet 8, Apendix B, Méthode OCTAVE Allegro). Elle comporte les informations suivantes :

  - quel est le bien ;
  - le bien est-il important ;
  - une description du bien ;
  - qui est responsable du bien ;
  - quels sont les besoins en sécurité du bien ?

- **Étape 3 : « Identify information asset containers »**

Cette étape permet d'identifier l'endroit où les biens de type information sont stockés. Cela peut aller d'une feuille de papier à un système d'information out-source.

La méthode met à disposition un template pour qualifier les containers. Deux grandes catégories y sont répertoriées : les containers internes et externes. Dans ce template, (Worksheet 9, Apendix B, Méthode OCTAVE Allegro) les containers sont caractérisés par une description de l'implémentation, des règles de sécurité qui lui sont appliquées ainsi que par le listing des personnes autorisées à avoir accès à ces informations.

- **Étape 4 : "Identify areas of concern"**

Le but de cette étape est d'identifier pour chaque bien un certain nombre de vulnérabilités. Chaque vulnérabilité est caractérisée par (Worksheet 10, Apendix B, Méthode OCTAVE Allegro) :

  - un acteur ;
  - ce que va faire l'acteur ;
  - une motivation ;
  - l'effet sur le système ;
  - le besoin en sécurité ;
  - une probabilité ;
  - une conséquence.

- **Étape 5 : « Identify threat scenarios »**

L'objectif de cette étape est de reprendre les vulnérabilités et d'étendre leur étude afin d'imaginer des scénarios sur base des propriétés de ces vulnérabilités. Une représentation graphique sous forme d'arbre est utilisée pour mettre en forme les différents scénarios.

- **Étape 6 : « Identify risks »**

Le but de la sixième étape est d'établir les conséquences de la série de scénarios qui ont été déterminés lors l'étape précédente. Ces conséquences donnent les deux informations constituant un risque : une condition et un impact.

- **Étape 7 : « Analyze risks »**

Dans cette étape, il s'agit d'établir un score pour chaque risque trouvé dans l'étape précédente. Ce score donne un poids au risque. Celui-ci permet de comparer les risques entre eux.

Pour chaque critère de risque sélectionné à la première étape, nous évaluons ainsi l'impact du scénario. Le poids est obtenu en additionnant les différentes mesures par critère de sécurité.

- **Étape 8 : « Select mitigation approach »**

Au début cette étape, les risques sont déjà identifiés et ils ont tous obtenu un score. Ce score nous aide à choisir une stratégie dans le traitement de chaque risque. Il est alors possible soit d'annuler le risque par la mise en place de contre-mesures permettant de réduire le risque ou de le transférer, soit d'accepter le risque.

4.5 Choix d'une méthode

Afin d'aider les entreprises à choisir parmi toutes les méthodes disponibles, l'agence nationale de la sécurité informatique tunisienne a établi une liste de différents critères repris ci-dessous. [ANSI, 2014 p 2]:

- « La langue de la méthode ;
- la culture du pays d'où vient la méthode ;
- la base de connaissances et les outils mis à disposition par la méthode ;
- la documentation et la qualité de celle-ci ;
- la pérennité de la méthode ;
- la compatibilité avec les normes internationales ;
- le retour d'expériences. »

4.5.1 Comparaison entre les méthodes

Le tableau ci-dessous croise les différentes méthodes présentées précédemment avec les critères développés par l'agence nationale de la sécurité informatique tunisienne :

	EBIOS	**Méhari**	**OCTAVE**	**OCTAVE Allegro**
Langue	Français	Français	Anglais	Anglais
Origine	France	France	États-Unis	États-unis
Base de connaissances et outils mis à disposition par la méthode	Outil libre disponible	Outil à base de fichiers Excel.	Outil payant	Template de work sheet à compléter
La documentation et sa qualité	Méthodologie complète mise à disposition avec différentes études de cas réalisées afin d'aider à sa mise en place	Méthodologie complète mise à disposition	Méthodologie et documentation complètes disponibles gratuitement mais formation payante	Méthodologie gratuite et disponible mais formation payante
Pérennité de la méthode	Dernière mise à jour en 2010	Dernière mise à jour en 2010	Publiée en 2003	Publiée en 2007
Compatibilité avec les normes internationales	ISO 31000, 27001, 27005	ISO 27001, 27002, 27005	ISO 31010	N.A.
Retour d'expériences	Mise à disposition d'études de cas ainsi qu'existence d'un club d'utilisateurs	Méthode basée sur des feuilles Excel, pas simple à mettre en place car demande une adaptation de la base de connaissances avant sa mise en pratique	Mise en place pour une analyse globale d'une entreprise, doit être menée par une équipe interdisciplinaire	Focus sur l'information, toute l'entreprise doit être concernée

Tableau 1 : Croisement des différentes méthodes présentées avec les critères de l'ANSI [ANSI, 2014]

4.5.2 Choix de la méthode

En fonction des critères de sélection mis en avant précédemment, le choix de la méthode pour la suite de l'analyse s'est porté sur EBIOS. En effet, cette méthode présente les qualités suivantes :

- La langue : le Français étant ma langue maternelle.
- Origine de la méthode : elle a été rédigée par l'agence nationale française de la sécurité des systèmes d'information. Mon expérience professionnelle a été principalement acquise au sein d'entreprises belges et françaises.
- Base de connaissances et outils mis à disposition par la méthode : l'outil est disponible gratuitement.
- Documentation : la méthode met à disposition une documentation complète qui est appuyée par un exemple au fils des étapes, ce qui facilite la mise en pratique. En cas de difficultés liées à la compréhension, il est possible de se référer à l'exemple afin de valider la démarche.
- Pérennité : la méthode a subi une remise à niveau en 2010 et en 2011, une nouvelle étude de cas est sortie.
- Le retour d'expériences : mise à disposition d'études de cas et existence d'un club EBIOS.

4.6 Approfondissement de la méthode EBIOS

La méthode EBIOS est composée de cinq modules. Chacun de ceux-ci va être présenté de manière détaillée [SGDN / ANSSI / ACE / BAC, 2010] afin de pouvoir appliquer la méthode lors de l'analyse des différents patterns présentés dans la cinquième partie.

4.6.1 1er Module : la contextualisation

Ce premier module a pour objectif de collecter un maximum d'informations afin de garantir la mise en œuvre de la méthode. C'est dans cette partie que sont formalisés le contexte ainsi que le cadre dans lequel l'analyse sera effectuée. Ce module est composé de trois étapes. Chacune d'elles est subdivisée en différentes activités.

1) Définir le cadre de la gestion

a. ***Cadrer l'étude des risques :*** définir l'objectif de l'analyse ainsi que ce qui en est attendu en termes de documentation, et justifier des choix à poser par l'auditeur.
b. ***Décrire les contextes généraux :*** définir d'une part le contexte dans lequel l'analyse aura lieu et d'autre part ce qui constitue un risque pour l'organisation. Cette description permet d'intégrer au mieux l'analyse de gestion des risques au sein de la culture de l'entreprise, de sa structure et de son processus organisationnel.
c. ***Délimiter le périmètre de l'étude :*** définir un périmètre pour l'analyse.
d. ***Identifier les paramètres à prendre en compte :*** identifier les contraintes à prendre en compte lors de l'analyse.
e. ***Identification des sources de menaces :*** identifier les différentes sources de menaces qui pèseront sur les cas qui sont présentés. Ces sources sont classées dans deux catégories : externes et internes.

2) Préparer les métriques

a. ***Définir les critères de sécurité et élaborer les échelles de besoins :*** définir les critères de sécurité qui ont été retenus dans la contextualisation de l'étude et présenter ces critères sous forme d'échelle.

Par exemple, la **confidentialité** qui est un critère de sécurité où l'utilisateur n'a accès au bien que s'il y est autorisé.

Tableau 2 : Description des critères de confidentialité.

Niveau de l'échelle	**Description du niveau de confidentialité**
Public	Il n'y a pas de restriction de confidentialité sur le bien
Limité	Le bien est accessible au propriétaire ainsi qu'à un groupe restreint
Privé	Le bien ne peut être accessible qu'au propriétaire

b. ***Élaborer une échelle de niveau de gravité :*** une échelle définit chaque niveau de gravité.

Par exemple, la **gravité** qui est une échelle utilisée pour estimer la gravité d'un évènement redouté et des risques. Cela aura un impact sur la survie ou non de l'entreprise concernée.

Tableau 3 : Description des niveaux de gravité.

Niveau de l'échelle	Description des niveaux de gravité
Négligeable	L'entreprise surmontera les impacts sans difficulté
Limité	L'entreprise surmontera les impacts avec quelques difficultés
Important	L'entreprise surmontera l'impact avec de grosses difficultés
Critique	L'entreprise ne surmontera pas les impacts

c. ***Élaborer une échelle de niveau de vraisemblance :*** définir la vraisemblance d'une occurrence d'un risque et la placer sur une échelle.
Par exemple, la **vraisemblance** qui est une échelle utilisée pour estimer la vraisemblance d'un scénario de menaces et des risques.

Tableau 4 : Description des niveaux de vraisemblance

Niveau de l'échelle	Description des niveaux de vraisemblance
Minime	Cela ne devrait pas se (re)produire
Significatif	Cela pourrait se (re)produire
Fort	Cela devrait se (re)produire un jour ou l'autre
Maximal	Cela va certainement se (re)produire prochainement

d. ***Définir les critères de gestion des risques :*** définir la façon dont les évènements sont estimés et définir l'évaluation des risques.

3) Identifier les biens

a. ***Identifier les biens essentiels, leurs relations et leurs dépositaires :*** identifier les biens immatériels ayant de l'importance pour la société et qui doivent être protégés.
b. ***Identifier les biens supports, leurs relations et leurs propriétaires :*** identifier les biens matériels faisant partie du système d'information.
c. ***Déterminer le lien entre les biens essentiels et les biens supports :*** déterminer ce lien afin d'identifier quel bien support doit être considéré comme critique.
d. ***Identifier les mesures de sécurité existantes :*** lister l'ensemble des mesures de sécurité déjà mises en place.

4.6.2 2ème Module : l'étude des évènements redoutés

L'objectif de ce module est « *d'estimer les évènements redoutés pour chaque critère de sécurité et chaque bien essentiel identifié* » [SGDN / ANSSI / ACE / BAC, 2010]. Ainsi pour chacune des caractéristiques présentées dans le contexte, une liste d'évènements pouvant compromettre nos critères de sécurité est dressée. Ce module est subdivisé en différentes activités.

1) **Analyser tous les évènements redoutés**
   Pour chaque critère de sécurité sélectionné dans le contexte, un listing des évènements redoutés est dressé. Pour chacun de ceux-ci, un impact potentiel est attribué et sa source de menace est identifiée.

2) **Evaluer chaque évènement redouté**
   Chaque évènement redouté est évalué et on juge sa gravité dans le cas où l'évènement viendrait à être rencontré. L'importance de la gravité est évaluée à l'aide du tableau présentant des définitions dans le premier module au point « 4.6.1 ».

   « Pour estimer la gravité des évènements redoutés au cas où ceux-ci se réaliseraient, il convient d'attribuer un niveau de gravité à chaque évènement redouté en utilisant l'échelle de gravité définie.

   L'estimation est faite au regard :

   - de la valeur du bien essentiel considéré ;
   - de la hauteur et du nombre d'impacts identifiés.

   Elle ne doit pas tenir compte des éventuelles mesures de sécurité existantes. » [SGDN / ANSSI / ACE / BAC, 2010].

4.6.3 3ème Module : étude des scénarios

L'objectif de ce module est d'« *identifier les scénarios de menaces pour chaque critère de sécurité et chaque bien support identifié, et de les estimer en termes de vraisemblance »* [SGDN / ANSSI / ACE / BAC, 2010] afin de pouvoir choisir les contre-mesures à appliquer et de définir les priorités à y associer. La présentation de ces scénarios peut être exprimée sous forme de textes, d'arbres de menaces ou encore de grilles détaillées.

1) Analyser tous les scénarios de menaces

Un scénario est composé de trois éléments : le risque de la menace, la vulnérabilité et la source de la menace. Le scénario décrit comment la vulnérabilité peut être exploitée par la source de menaces.

Le processus d'élaboration des scénarios se concentre sur les menaces qui ont un lien avec les critères de sécurité désignés au sein du contexte d'analyse. Pour chaque menace, il y a une série de scénarios qui tient compte des différentes sources de menaces.

2) Évaluer chaque scénario de menace

Une fois les scénarios listés, une source et une pondération leur sont assignées afin que le scénario ait une chance d'aboutir. Pour donner une vraisemblance au scénario, il faut utiliser l'échelle définie dans le premier module au point « 4.6.1 ».

« Pour estimer la vraisemblance des scénarios de menaces, il convient d'attribuer un niveau à chaque scénario de menace en utilisant l'échelle de vraisemblance définie. L'estimation est essentiellement faite au regard :

- de l'existence plus ou moins avérée et de la facilité d'exploitation des vulnérabilités identifiées ;
- de l'exposition aux menaces considérées ;
- de l'exposition et du potentiel des sources de menaces identifiées.

Elle ne doit pas tenir compte des éventuelles mesures de sécurité existantes. » [SGDN / ANSSI / ACE / BAC, 2010].

4.6.4 4ème Module : étude des risques

Selon EBIOS, « L'étude des risques *a pour objectif de mettre en évidence de manière systématique les risques pesant sur les périmètres de l'étude, puis de choisir la manière de les traiter en tenant compte des spécificités du contexte.* » [SGDN / ANSSI / ACE / BAC, 2010].

Cette étape suit l'analyse des menaces et l'analyse des scénarios. En effet, elle croise les données récoltées lors des deux étapes précédentes afin de mettre en évidence les risques définis comme tels.

« Il y a un risque de sécurité de l'information dès que l'on a conjointement :

- une source de menace ;
- une menace ;
- une vulnérabilité ;
- un impact.

On peut ainsi comprendre qu'il n'y a plus de risque si l'un des facteurs manque. » [SGDN / ANSSI / ACE / BAC, 2010].

1) Apprécier les risques

a. Analyser les risques

Dans cette partie, les données obtenues dans les deux modules précédents sont confrontées. Les valeurs de vraisemblance et de gravité sont réévaluées en fonction des mesures de sécurité déjà existantes. Un risque est composé d'un évènement redouté ainsi que de tous ses scénarios.

b. Evaluer les risques

Un risque est mis en avant par son évènement redouté et tous les scénarios qui y sont rattachés. Quant à leur évaluation en tant que risque, elle est exécutée sur base des conséquences et de la probabilité que le risque se réalise. Il est modélisé par un tableau à double entrée qui présente d'une part, les risques d'occurrence et d'autre part, le niveau de gravité.

2) Identifier les objectifs de sécurité

a. Choisir les options du traitement des risques

Dans cette étape, il faut définir comment seront traités les risques identifiés.

Il existe plusieurs façons de faire face à un risque.

- **L'éviter** : on change le contexte pour annuler le risque par des nouvelles mesures de sécurité ; le risque ne fera plus partie des risques résiduels.
- **Le réduire** : on prend un certain nombre de mesures afin de réduire la probabilité d'une occurrence ; le but étant de réduire le nombre de scénarios de menace à un niveau acceptable ; le risque résiduel est toujours présent au sein du rapport mais le risque est acceptable.
- **L'accepter** : on notifie que le risque est acceptable ; le risque reste au sein de la liste des risques résiduels comme tel.

- **Le transférer** : cas typique d'un outsourcing, on ne prend pas la responsabilité du risque et de ce qu'il implique au point de vue du contexte, et ce sur base contractuelle avec une indemnité en cas de non-respect ; cela équivaut aux principes des assurances ; le risque disparait définitivement du listing de risques résiduels.

b. Analyser les risques résiduels

Dans cette étape, le but est d'identifier les risques résiduels pour lesquels un risque subsiste après le choix du traitement à effectuer. Il faudra alors évaluer la vraisemblance et la gravité des risques résiduels.

4.6.5 5ème Module : étude des mesures de sécurité

Selon EBIOS, « l'étude des mesures de sécurité a pour objectif de déterminer les moyens de traiter les risques et de suivre leur mise en œuvre, en cohérence avec le contexte de l'étude. Les réflexions sont préférentiellement menées de manière conjointe entre les niveaux fonctionnels et techniques. » [SGDN / ANSSI / ACE / BAC, 2010].

1) Formaliser les mesures de sécurité à mettre en œuvre

a. Déterminer les mesures de sécurité

Dans cette action, il faut définir les moyens qui vont être mis en place afin d'atteindre les objectifs de sécurité qui ont été fixés. Il y aura ainsi trois catégories de mesures : celle visant à prévenir le risque, celle visant la protection de l'élément vulnérable et celle visant à minimiser l'impact une fois qu'un incident survient.

b. Analyser les risques résiduels

Cette analyse permet d'identifier et d'évaluer les risques résiduels après la mise en place de mesures définies précédemment.

c. Etablir une déclaration d'applicabilité

Il s'agit de faire référence aux contraintes et aux hypothèses émises dans la contextualisation et de justifier si elles ont été prises en compte ou non dans l'étude.

2) Mettre en œuvre les mesures de sécurité

a. Elaborer le plan d'action et suivre la réalisation des mesures de sécurité

Il s'agit de planifier les actions à effectuer en vue de mettre en place des mesures de sécurité. Les mesures qui sont déjà en place ne doivent pas faire l'objet d'une planification. Quant aux nouvelles, elles devront être planifiées.

b. Analyser les risques résiduels

L'analyse des risques résiduels réels a lieu une fois que les mesures définies ci-dessus sont en place.

c. Prononcer l'homologation de sécurité

Il s'agit de faire valider de façon formelle la sécurisation par le biais d'une homologation.

5 Application d'EBIOS aux patterns du cloud

Dans ce chapitre, nous procéderons à une analyse de différents patterns de développement liés au cloud à partir de la méthode EBIOS. Les patterns sont des solutions génériques répondant à un certain nombre de problèmes liés aux propriétés du cloud. Les patterns retenus sont issus du livre «Cloud Computing Patterns » [Fehling, 2014].

Dans un premier temps, nous décrivons le contexte dans lequel l'analyse va être effectuée. Nous analysons ensuite chaque pattern en suivant la méthode EBIOS. Pour chaque pattern, les évènements redoutés sont listés et évalués à l'aide de l'échelle définie dans le contexte. Sur la base des évènements redoutés, des scénarios sont élaborés. Ils seront par la suite analysés selon une échelle, définie elle aussi, dans le contexte.

L'évènement redouté et les scénarios développés au niveau de chaque pattern sont déterminés de deux manières différentes : soit sur base d'expérimentations réalisées dans une entreprise ayant des caractéristiques identiques à celle décrite dans la contextualisation, soit à partir de la littérature scientifique traitant les points de sécurité à prendre en compte lors du développement d'applications.

5.1 Etude du contexte

5.1.1 Cadrage de l'étude des risques

L'étude porte sur la mise en évidence des risques liés à différents patterns de développement à destination du cloud.

Les trois patterns architecturaux retenus dans cette étude ont été expérimentés en entreprise. Il s'agit des patterns suivants : « two-tier cloud application », « three-tier cloud application » et « hybrid development environment ». Chacun de ces trois patterns est également composé de différents patterns : «stateful component », « stateless component », « user interface component », « elastic load balancer », « message mover », « processing component » et « data access component ». Comme ceux-ci ne sont pas systématiquement repris dans chaque pattern architectural, ils seront analysés de manière indépendante.

5.1.2 Description du contexte général

Cette recherche fait référence à une société de services informatiques. Celle-ci a un portefeuille de clients de plus en plus exigeants en termes de prix liés à la prestation mais aussi en termes de qualité attendue. La sécurité au niveau du service est une de leurs priorités. En effet, leur application traite de données confidentielles relatives aux clients (données financières, médicales, …) ; leur confidentialité ainsi que leur intégrité sont requises. D'autre part, le niveau de disponibilité du service doit être assuré à hauteur d'un SLA[13] de 99,9 % up-time.

Cette société veut s'orienter vers le cloud computing afin de réduire ses couts d'exploitation et de profiter de la flexibilité du modèle. Dès lors, dans le cadre d'application monolithique, la migration ne se fait pas telle quelle. Une adaptation de l'architecture des services doit être établie, ce qui fait appel aux patterns du livre cité préalablement.

Afin de répondre aux demandes du client, une analyse méthodique des différents patterns est menée. Elle tient compte des critères suivants : intégrité, confidentialité et disponibilité des systèmes des clients.

5.1.3 Délimitation du périmètre de l'étude

Il s'agit des trois patterns architecturaux définis dans le cadrage de l'étude des risques au point « 5.1.1

5.1.4 Identification des paramètres à prendre en compte

Les paramètres sont des contraintes dont il faut tenir compte lors de l'analyse. Il existe différents types de contraintes. Dans le contexte de cette étude, seules les contraintes d'ordre technique et politique sont prises en compte.

Contraintes techniques :

- deux data-centers ;
- applications distribuées ;
- lien avec des partenaires externes, plusieurs niveaux de réseau.

[13] Service Level Agreement

Contraintes politiques :

- « release » à date fixe ;
- confidentialité totale de certaines données.

5.1.5 Identification des sources de menace

Les différentes menaces qui pèsent sur les cas choisis sont identifiées en fonction des sources. Celles-ci sont classées dans deux catégories : externes et internes. Cette identification provient d'expérimentations menées au sein de deux grandes entreprises.

Externes :

- hacker malveillant (intentionnel) ;
- utilisateur lambda (utilisation normale) ;
- condition cloud provider ;
- certificat autorité.

Internes :

- développeur ;
- administrateur ;
- service provider ;
- infrastructure.

Ce listing tient compte du contexte et de la cible qui vont être analysés. Dans ce cas-ci, l'analyse porte sur des patterns de développement et donc il y a peu d'intérêt à ajouter des sources de menaces telles qu'une panne électrique ou bien une catastrophe naturelle. Ces sources sont à prendre en compte dans le cas d'analyse de risques d'infrastructure.

5.1.6 Identification des besoins en termes de sécurité et élaboration des échelles de besoin

Les critères de sécurité déjà sélectionnés dans la contextualisation sont la confidentialité, l'intégrité et la disponibilité. Voici leur définition ainsi que leur niveau d'échelle qui est défini par la méthode et qui peut être revu en fonction des attentes du client.

- **Confidentialité** : critère de sécurité où l'utilisateur n'a accès qu'au bien où celui-ci est autorisé. La définition et le niveau d'échelle sont définis par la méthode.

Tableau 5 : Description des critères de confidentialité

Niveau de l'échelle	Description du niveau de confidentialité
Public	Il n'y a pas de restriction de confidentialité sur le bien.
Limité	Le bien est accessible au propriétaire ainsi qu'à un groupe restreint.
Privé	Le bien ne peut être accessible qu'au propriétaire.

- **Intégrité** : critère de sécurité garantissant l'intégrité du bien afin qu'il ne soit pas altéré par un tiers. La définition et le niveau d'échelle sont définis par la méthode.

Tableau 6 : Description des critères d'intégrité

Niveau de l'échelle	Description du niveau de l'intégrité
Détectable	Le bien essentiel peut ne pas être intègre si l'altération est identifiée.
Maitrisé	Le bien essentiel peut ne pas être intègre, si l'altération est identifiée et l'intégrité du bien essentiel retrouvée.
Intègre	Le bien essentiel doit être rigoureusement intègre.

- **Disponibilité** : critère de sécurité qui va garantir la disponibilité d'un bien. Les définitions sont issues de la méthode, par contre le niveau de l'échelle a été adapté en fonction de l'entreprise concernée par l'analyse avec comme objectif d'atteindre le SLA de 99,9% up-time.

Tableau 7 : Description des critères de disponibilité

Niveau de l'échelle	Description du niveau de disponibilité
De 24 à 72h	Le bien essentiel peut être indisponible entre 24 et 72 heures.
De 4 à 24h	Le bien essentiel peut être indisponible entre 4 et 24 heures.
De 1 à 4h	Le bien essentiel peut être indisponible entre 1 et 4 heures.
Moins de 1 h	Le bien ne peut en aucun cas être indisponible plus de 1 heure.

5.1.7 Définition de l'échelle de gravité

- **Gravité** : échelle utilisée pour estimer la gravité d'un évènement redouté et des risques qui ont un impact sur la survie ou non de l'entreprise concernée. L'échelle est définie par la méthode.

Tableau 8 : Description des niveaux de gravité

Niveau de l'échelle	Description des niveaux de gravité
Négligeable	L'entreprise surmontera les impacts sans difficulté.
Limité	L'entreprise surmontera les impacts avec quelques difficultés.
Important	L'entreprise surmontera l'impact avec de grosses difficultés.
Critique	L'entreprise ne surmontera pas les impacts.

5.1.8 Définition de l'échelle de vraisemblance

- **Vraisemblance** : échelle utilisée pour estimer la vraisemblance d'un scénario de menace et des risques. L'échelle est issue de la méthode.

Tableau 9 : Description des niveaux de vraisemblance

Niveau de l'échelle	Description des niveaux de vraisemblance
Minime	Cela ne devrait pas se (re)produire.
Significatif	Cela pourrait se (re)produire.
Fort	Cela devrait se (re)produire un jour ou l'autre.
Maximal	Cela va certainement se (re)produire prochainement.

5.1.9 Etude des risques

Une fois le niveau établi pour un évènement et un scénario, il est possible de placer le risque sur ce tableau à double entrée. Il s'agit du modèle proposé par la méthode.

Tableau 10 : Tableau d'analyse gravité vs vraisemblance

Gravité	4. Critique				
	3. Importante				
	2. Limitée				
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

Les résultats obtenus donnent une classification des risques selon trois catégories.

Risques négligeables	Risques significatifs	Risques intolérables

5.1.10 L'identification des biens

Dans le périmètre de l'étude, nous avons délimité l'analyse de risques sur la base de patterns de développement du cloud. C'est à partir de ce périmètre que les biens vont être identifiés, l'infrastructure ne faisant pas partie du contexte. Il existe deux sortes de biens : les biens essentiels et les biens supports.

Biens essentiels :

- les données ;
- les sessions utilisateurs ;
- le système de messaging (« message-oriented middleware »).

Biens supports :

- le code d'application (front-end, back-end, data-access) ;
- l'infrastructure ;
- les fichiers de configuration ;
- les certificats.

Les biens identifiés ont un lien direct avec l'applicatif car le but est d'analyser le risque au niveau du pattern.

Tableau 11 : Lien entre les biens essentiels et les biens supports.

	Données	**Session utilisateur**	**Messages**
Code applicatif	X	X	X
Infrastructure	X	X	X
Fichier de configuration			X
Certificat		X	

5.1.11 L'identification des mesures de sécurité existantes

Dans ce cas, comme le système d'information n'est pas encore construit, il n'existe pas encore de mesures de sécurité.

5.2 Analyse des Patterns

5.2.1 Méthodologie

Pour faciliter la compréhension du lecteur, les modules 2, 3 et 4 de la méthode EBIOS ont été intégrés dans l'analyse de chaque pattern. Chaque analyse de pattern aura la structure suivante.

A. Présentation

La présentation de chaque pattern est accompagnée d'un schéma issu du livre « Cloud Computing Pattern » [Fehling, 2014].

B. Évènements redoutés applicables

Il y a une section reprenant les évènements redoutés (Module 2). Une parenthèse a été ajoutée après chaque évènement redouté afin d'indiquer s'il s'agit d'un évènement expérimenté entreprise ou d'un fait issu de la littérature.

C. Évaluation des évènements redoutés

Ces évènements sont analysés afin d'attribuer une source de menace ainsi qu'un niveau de gravité en cas d'occurrence.

D. Scénarios par évènement

Ensuite, une série de scénarios est développée pour chaque évènement et chaque source de menace.

E. Évaluation des scénarios

Chaque scénario est évalué afin de lui attribuer une vraisemblance. L'attribution des cotes à savoir la gravité ou bien la probabilité est fondée d'une part sur mon expérimentation au sein de la production de deux grandes entreprises de services informatiques et d'autre part,

sur la base des échelles définies dans le contexte. Le but de l'utilisation de ces échelles est de garantir une uniformité dans l'analyse.

F. Analyse des risques

Il s'agit de présenter la gravité et la vraisemblance dans le tableau à double entrée proposé dans l'étude des risques au point « 5.1.9 ». Ce tableau qualifie ainsi le risque.

5.2.2 Stateful Component

A. Présentation

La solution proposée par ce pattern est de stocker un certain nombre de données de navigation (session) client au sein même des instances applicatives. Ces données sont synchronisées entre les différentes instances, ce qui permet au client de pouvoir se connecter sur n'importe quelle instance sans perdre sa session.

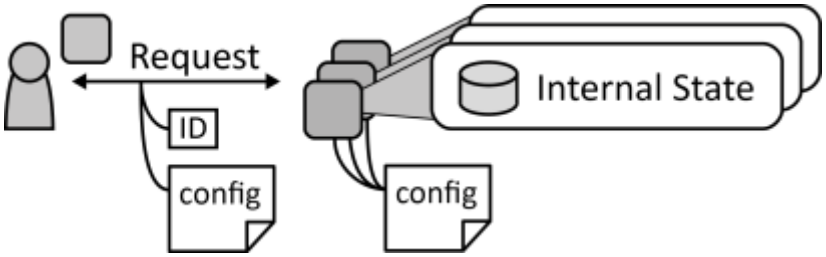

Figure 6 : Stateful components (Fehling, 2014, p. 169)

B. Évènements redoutés applicables

1. Perte d'intégrité des données stockées entre les différentes instances déployées (expérimentation).
1. Indisponibilité du crash d'une instance, perte de données session (expérimentation).
2. Taille d'objets trop grande et donc pas assez de mémoire (expérimentation).
3. Requête bloquée (expérimentation).
4. Vol de données [WASC, 2010].

C. Evaluation des évènements redoutés

Tableau 12 : Evènements redoutés stateful components

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	Intègre	- Administrateur (faute) - Infrastructure	- Perte de la session	Limité
2	Intègre	- Développeur - Infrastructure	- Perte de données non persistées	Important
3	Moins de 1h	- Infrastructure - Utilisateur - Hacker	- Perte de l'instance	Limité
4	Détectable	- Utilisateur - Hacker	- Peut bloquer la requête	Limité
5	Détectable	- Hacker - Administrateur	- Fuite de données dans le domaine public	Critique

D. Scénarios par évènement

1. ***Intégrité des données stockées entre les différentes instances déployées***
   - **1.1.** Mauvaise configuration des instances qui induit l'impossibilité d'une synchronisation entre le contenu des différentes instances. Quand un utilisateur passe d'une instance à une autre, il ne retrouve pas ses données de session.
   - **1.2.** Dans le cas où le nombre d'instances est trop grand, le trafic de synchronisation va lui aussi s'accroitre. Une congestion peut ainsi apparaitre entrainant une impossibilité de synchronisation. L'utilisateur peut donc se connecter sur une des instances qui n'a pas ses données de session à jour.

2. ***Indisponibilité due au crash d'une instance, perte de données session***
   - **2.1.** L'instance s'arrête, car elle est surchargée d'utilisateurs. Elle n'arrive plus à répondre.
   - **2.2.** L'instance est arrêtée pour maintenance par un administrateur.

3. ***Taille d'objets trop grande et donc pas assez de mémoire***
   - **3.1.** L'application gère mal les informations stockées au sein de l'instance. Les objets ne sont pas assez rapidement persistés. L'instance s'arrête.
   - **3.2.** La cache de l'application n'est pas gérée correctement. L'instance s'arrête.
   - **3.3.** Le nombre de requêtes explose, l'utilisation explose ou bien l'application subit une attaque DDOS. L'instance s'arrête.

4. ***Requête bloquée***
   - **4.1.** Soumission d'une requête avec des données provoquant un dead-lock. L'instance ne peut plus accepter de requêtes.
   - **4.2.** Transaction distribuée, non gérée par le code.
   - **4.3.** Attaque de type DDOS trop grande pour être gérée.

5. ***Vol de données***
   - **5.1.** Par manipulation de cookies.
   - **5.2.** Par attaque de « brute force ».
   - **5.3.** A l'aide de « Cross-Site Scripting ».
   - **5.4.** Un administrateur vole les données auxquelles il peut avoir accès au sein même de la société.

E. Evaluation des scénarios

Tableau 13 : Scénario stateful components

Scénario de menaces	Source de menaces	Vraisemblance
1.1	- Administrateur	Significative
1.2	- Utilisateur	Significative
2.1	- Utilisateur	Significative
2.2	- Administrateur	Significative
3.1	- Développeur	Minime
3.2	- Développeur	Significative
3.3	- Utilisateur - Hacker	Forte
4.1	- Développeur	Minime
4.2	- Développeur	Significative
4.3	- Hacker	Forte
5.1	- Hacker	Significative
5.2	- Hacker	Significative
5.3	- Hacker	Significative
5.4	- Administrateur	Minime

F. Analyse des risques

Tableau 14 : Risques stateful components

Gravité	4. Critique	5.4	5.1 ; 5.2 ; 5.3		
	3. Importante		2.1 ; 2.2		
	2. Limitée	3.1 ; 4.1	1.1 ; 1.2 ; 3.2 ; 4.2	3.3	4.3
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.3 Stateless Component

A. Présentation

Cette solution propose que l'état de l'application soit géré à l'extérieur des composants applicatifs. L'application peut se baser sur plusieurs méthodes pour conserver un état, soit par le biais d'une base de données dans laquelle toutes les informations de navigation sont persistées, soit via l'utilisation de la session de l'utilisateur, ainsi l'état est du côté client. A chaque requête, celui-ci doit fournir les informations utiles. L'utilisation des deux en parallèle est également possible.

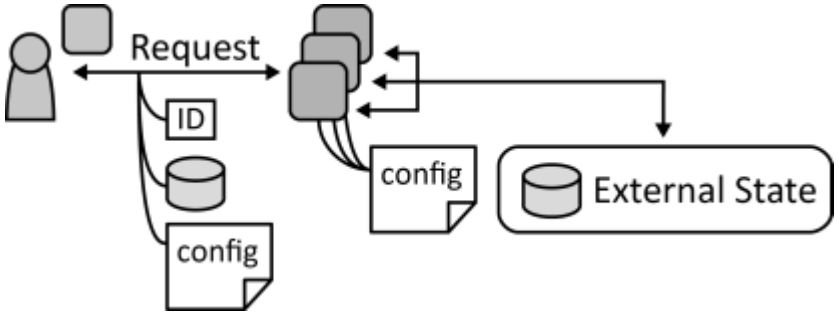

Figure 7: Stateless components (Fehling, 2014, p. 172)

B. Evènements redoutés applicables

1. Problème de performance de la base de données (expérimentation).
2. Session spoffing [WASC, 2010].
3. Requêtes mal formées [WASC, 2010].
4. Intégrité de configurations (expérimentation).
5. Vol de données [WASC, 2010].

C. Evaluation des évènements redoutés

Tableau 15 : Evènements redoutés stateless components

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	Moins de 1h	- Utilisateur - Hacker - Infrastructure	- Performance - Indisponibilité - Perte de clients	Important
2	Intègre	- Hacker	- Pertes de données clients - Perte de clients	Important
3	Détectable	- Hacker - Utilisateur	- Erreur pour l'utilisateur	Négligeable
4	Intègre	- Administrateur - Hacker	- Nouvelle instance mal configurée	Critique
5	privé	- Hacker - Administrateur	- Fuite de données dans le domaine public	Critique

D. Scénarios par évènement

1. ***Problème de performance de la base de données***

1.1. Suite à un nombre croissant d'instances, le nombre de requêtes sur la base de données peut entrainer un problème de performance ou de disponibilité. La base de données est submergée par le nombre de requêtes.

2. ***Session spoffing***

2.1. Suite à une mauvaise configuration de la gestion des sessions, le hacker pourrait détourner la session d'une personne déjà connectée et ainsi récupérer les données du client voire même les prendre et agir au nom du client.

3. ***Requêtes mal formées***

3.1. Dans le cadre d'application type REST, les requêtes envoyées au composant peuvent être manipulées. Dans le cas où le composant ne tient pas compte de cette probabilité, il est possible que les données soient altérées.

4. ***Intégrité de configurations***

4.1. Un administrateur introduit une modification sans prévenir personne et impacte la stabilité de toutes les instances.

4.2. Le fichier de configurations est supprimé par une personne malveillante, ce qui impacte l'auto provisioning des instances.

5. ***Vol de données***

5.1. Utilisation d'une attaque « brute force » pour l'accès à des informations non autorisées.

5.2. Utilisation des « Cross Site Scripting » pour récupérer des informations sur le système ou bien les données.

5.3. Un administrateur vole les données depuis l'intérieur.

E. Evaluation des scénarios

Tableau 16 : Scénarios stateless components

Scénario de menace	Source de menace	Vraisemblance
1.1	- Utilisateur	Significative
2.1	- Hacker - Administrateur	Significative
3.1	- Hacker	Maximale
4.1	- Administrateur	Minime
4.2	- Hacker	Minime
5.1	- Hacker	Significative
5.2	- Hacker	Significative
5.3	- Administrateur	Minime

F. Analyse des risques

Tableau 17 : Risques stateless components

Gravité	4. Critique	4.1 ; 4.2 ; 5.3	5.1 ; 5.2		
	3. Importante		1.1 ; 2.1		
	2. Limitée				
	1. Négligeable				3.1
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.4 User Interface Component

A. Présentation

Dans le cadre de ce pattern, l'idée est de permettre à l'utilisateur d'interagir avec l'interface graphique et de la customiser de façon synchrone. Les actions sont envoyées au serveur par l'utilisation de messages asynchrones. Ce pattern est très utilisé afin de garantir le fait de ne pas être dépendant du service. L'interface peut être visuellement adaptée pour chaque utilisateur, et ce tant sur l'agencement que sur le contenu.

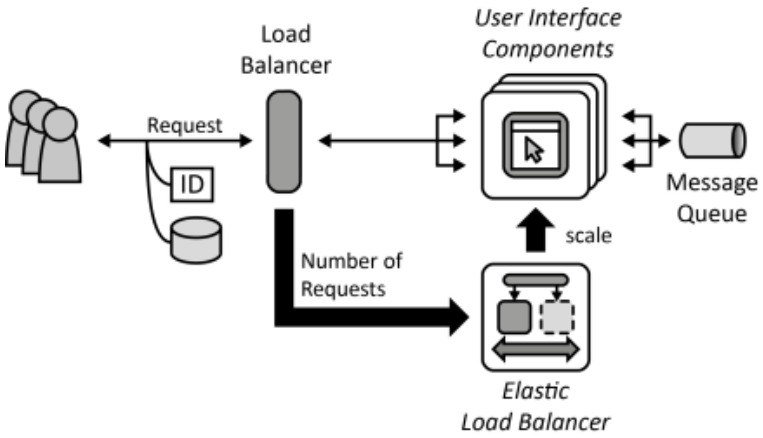

Figure 8 : User interface component (Fehling, 2014, p. 176)

B. Evènements redoutés applicables

1. Interface dynamique et complexe source de faille [SAFECode, 2011].
2. Augmentation du nombre de dépendances et donc probabilité plus importante qu'un composant tombe (expérimentation).
3. Utilisation d'une autre session ou d'une autre configuration (expérimentation).
4. Intégrité des messages asynchrones entre l'interface et le back-end [CSA, 2013].
5. Queue indisponible (expérimentation).
6. Perte de données [WASC, 2010].
7. Détournement de fonctionnalités [WASC, 2010].

C. Evaluation des évènements redoutés

Tableau 18 : Evènements redoutés user interface components

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	Intègre Privé	- Hacker - Utilisateur	- Accès à des données non autorisé	Important
2	Moins de 1h	- Administrateur - Infrastructure	- Indisponibilité de l'interface - Transaction bloquée	Important
3	Privé	- Hacker	- Accès au profil d'un tiers	Limité
4	Intègre	- Hacker	- Accès à des données interdites	Important
5	Moins de 1h	- Infrastructure - Administrateur	- Système non opérationnel	Critique
6	Privé	- Hacker - Administrateur	- Fuite de données dans le domaine public	Critique
7	Intègre	- Hacker	- Accès à d'autres composants - Perte d'informations	Critique

D. Scénarios par évènement

1. ***Interface dynamique et complexe pouvant être source de faille***
   - **1.1.** L'interface est composée de différents modules qui pourraient provenir de différents providers. Imaginons qu'un des modules composant l'interface contienne une faille de sécurité. Cette faille remet en cause la sécurité de toute l'application. Dans le cas où le module incriminé n'a pas été développé en interne, il est d'autant plus dur d'obtenir un correctif.
   - **1.2.** Dans le cadre d'une interface dont l'utilisateur est capable de soumettre lui-même le module, il est encore plus difficile d'avoir confiance dans ce code. Un hacker pourrait écrire un module et l'utiliser pour détourner les autres composants disponibles.

2. ***Augmentation du nombre de dépendances. Probabilité plus importante qu'un composant tombe***
   - **2.1.** L'application pourrait dans l'ensemble ne pas fonctionner si l'un des composants est arrêté.

3. ***Utilisation d'une autre session ou d'une autre configuration***
   - **3.1.** La configuration de l'application est mal faite ; le mapping entre les utilisateurs et leur interface peut ne pas fonctionner ; un utilisateur 'Y' se retrouvant avec l'interface de l'utilisateur 'X'.
   - **3.2.** Un hacker arrive à récupérer l'interface d'un autre utilisateur.

4. ***Intégrité des messages asynchrones entre l'interface et le back-end***
   - **4.1.** L'interface étant dynamique, une grosse partie du code sera chargée par le client. Cela implique que les données renvoyées au serveur puissent avoir été manipulées afin de by-passer certaines règles du métier.

5. ***Queue indisponible***
   - **5.1.** Il y a arrêt des queues car plus de mémoire.
   - **5.2.** Il y a arrêt des queues par le fournisseur de services ou l'administrateur.

6. ***Perte de données***
   - **6.1.** Il s'agit d'utiliser une attaque de « brute force » afin de récupérer des données qui n'étaient pas autorisées.
   - **6.2.** Utilisation du « Cross Site Scripting » afin de récupérer des informations cachées.

7. ***Détournement de fonctionnalités***
   - **7.1.** Utilisation d'un composant de l'interface dans le but d'accéder à d'autres composants suite à une authentification au sein du premier composant.
   - **7.2.** Il s'agit d'une utilisation bien connue par le détournement d'un composant de mails et l'envoi massif de mails.

E. Evaluation des scénarios

Tableau 19 : Scénarios user interface component

Scénario de menace	Source de menace	Vraisemblance
1.1	- Développeur - Administrateur	Significative
1.2	- Hacker - Utilisateur	Forte
2.1	- Infrastructure	Forte
3.1	- Développeur - Administrateur	Minime
3.2	- Hacker	Forte
4.1	- Utilisateur - Hacker	Maximale
5.1	- Infrastructure	Forte
5.2	- Administrateur	Significative
6.1	- Hacker	Significative
6.2	- Hacker	Significative
7.1	- Hacker	Significative
7.2	- Hacker	Forte

F. Analyse des risques

Tableau 20 : Risques user interface component

Gravité	4. Critique		5.2	5.1	
	3. Importante		1.1 ;	1.2 ; 2.1	4.1
	2. Limitée	3.1		3.2	
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.5 Elastic Load Balancer

A. Présentation

Ce type de composant a pour but de répondre à la demande des utilisateurs. Son fonctionnement est le suivant. A l'image d'un load-balancer réseau, il est chargé de dispatcher la charge des appels sur les différentes instances capables de prendre en charge la requête. Cependant, là où il se distingue, c'est par sa capacité à interagir avec l'infrastructure. Ce composant intègre en effet un certain nombre de règles afin de pouvoir prendre la décision de générer de nouvelles instances ou encore de démissionner de celles-ci. Ces règles sont basées sur le « capacity planning » de l'application. Ainsi tout appel entrant sera pris en charge dans les meilleures conditions.

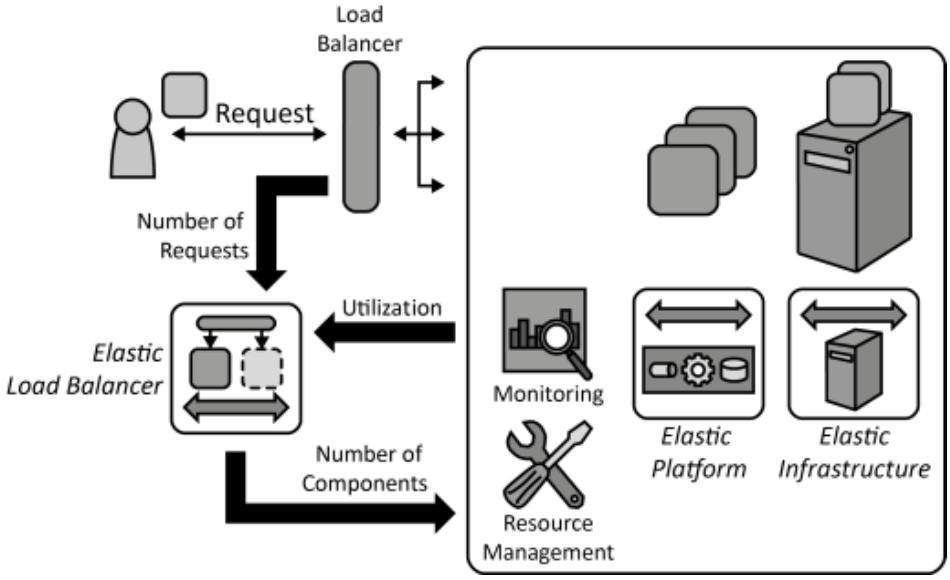

Figure 9 : Elastic load balancer interacting with an elastic platform or elastic infrastructure (Fehling, 2014, p. 255)

B. Evénements redoutés applicables

1. Concurrence mal gérée (expérimentation).
2. Limite de l'infrastructure dépassée (expérimentation).
3. Attaque sur le load-balancer (expérimentation).
4. Données du monitoring fausses (expérimentation).

C. Evaluation des évènements redoutés

Tableau 21 : Evènements redoutés elastic load balancer

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	- Intègre	- Développeur - Administrateur	-Transaction bloquée - Perte d'intégrité entre les instances	Important
2	- Moins de 1h	- Infrastructure - Utilisateur	- Indisponibilité	Limité
3	- Moins de 1h	- Hacker	- Indisponibilité	Limité
4	- Intègre	- Administrateur - Développeur - Infrastructure - Hacker	- Indisponibilité - Facturation trop importante	Limité

D. Scénarios par évènement

1. Concurrence mal gérée

1.1. Le développeur ne tient pas compte du fait d'avoir plusieurs instances applicatives qui doivent se partager la même ressource. Cela peut entrainer un blocage de transactions ou une incohérence dans le résultat des transactions.

1.2. L'administrateur ne configure pas correctement le/les transaction(s) manager, ce qui peut entrainer des problèmes d'intégrité entre les instances.

2. Limite de l'infrastructure dépassée

2.1. Du point de vue du développeur, l'utilisation de ce pattern donne l'impression d'avoir accès à une capacité d'infrastructure infinie. Mais malgré le fait de pouvoir provisionner des machines de façon indéfinie, il faut avoir une idée de la consommation d'une application par session afin de l'anticiper. Si l'administrateur ne fait pas ce travail en collaboration avec le développement, une indisponibilité peut arriver.

3. Attaque sur le load-balancer

3.1. Attaque de « Distributed Denial Of Service ».

3.2. Attaque de « SYN flood ».

3.3. Attaque « slowloris ».

3.4. Mauvais suivi des protocoles ssl (heard bleed).

4. Données du monitoring fausses

4.1. Soit la configuration du monitoring n'est pas correcte, ce qui donne de mauvaises informations au « load balancer » et les décisions qu'il prend, sur la base des règles préétablies, ne sont pas correctes. Soit les instances ne sont pas relâchées à temps, ce qui consomme des ressources ou pire dans le cas d'un besoin en termes de capacité, le load balancer ne crée pas de nouvelles instances rendant ainsi la plateforme indisponible.

4.2. Le monitoring ne donne plus d'informations car il est tombé en panne, ce qui aura des conséquences identiques au premier scénario.

E. Evaluation des scénarios

Tableau 22 : Scénarios elastic load balancer

Scénario de menaces	**Source de menaces**	**Vraisemblance**
1.1	- Développeur	Significative
1.2	- Administrateur	Significative
2.1	- Infrastructure	Significative
3.1	- Hacker	Forte
3.2	- Hacker	Significative
3.3	- Hacker	Significative
3.4	- Administrateur	Significative
4.1	- Administrateur	Significative
4.2	- Infrastructure	Significative

F. Analyse des risques

Tableau 23 : Risques elastic load banlancer

Gravité	4. Critique				
	3. Importante		1.1 1.2		
	2. Limitée		2.1 ; 3.2 ; 3.3 ; 3.4 ; 4.1 ; 4.2	3.1	
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.6 Message Mover

A. Présentation

Le message mover est utilisé dans le cadre d'applications distribuées, il fait appel au messaging. Ce pattern permet de déplacer des messages d'une queue vers une autre. Il permet donc d'établir le lien entre différents composants, voire environnements. C'est un système très utile afin de rendre transparente la communication entre différents composants applicatifs.

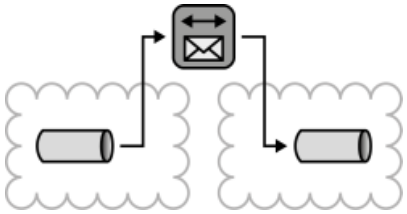

Figure 10 : Message mover integrating queues of two environments (Fehling, 2014, p. 226)

B. Evènements redoutés applicables

1. Perte de messages (expérimentation).
2. Arrêts du système (impact sur toute la chaine) (expérimentation).
3. Traitement de messages plusieurs fois (expérimentation).
4. Messages corrompus [CSA, 2013].
5. Détournement des messages volontaires ou involontaires (connexion à la mauvaise queue) (expérimentation).
6. Mauvaise gestion des certificats (expérimentation).
7. Injection de messages non autorisés [CSA, 2013].
8. Plus d'espace (expérimentation).

C. Evaluation des évènements redoutés

Tableau 24 : Evènements redoutés message mover

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	Intégrité	- Infrastructure - Administrateur	- Perte de données client	Important
2	Moins de 1h	- Infrastructure -Administrateur	- Quelques destinations ne sont plus approvisionnées -La queue source va grossir	Important
3	Détectable	- Développement	-Traitements multiples -Blocage de transactions -Violation de contraintes	Critique
4	Intègre	- Hacker - Développeur	-Blocage de transactions -Violation de contraintes	Critique
5	Intègre	- Administrateur	- Perte de messages - Blocage du système - Intégrité des bases de données en arrière-plan	Critique
6	Moins 1h	- Administrateur - Développeur	- Arrêt du système	Critique
7	Détectable Privé	-Hacker	-Blocage du système -Fuite de données -Intégrité des données en aval	Critique
8	Moins 1h	- Infrastructure	- Transfert et persistance de messages impossibles	Critique

D. Scénarios par évènement

1. Perte de messages

1.1. Le disque dur de destination est plein, il n'est plus possible de persister le message au sein de la queue.

1.2. Dans le cadre d'une connexion UDP, la perte de paquet réseaux est possible.

1.3. Il y a une mauvaise configuration des queues qui entraine une perte de messages.

2. ***Arrêt du système***
   - **2.1.** Le processus de transfert est arrêté par une mauvaise manipulation ou une surcharge.

3. ***Traitement d'un même message plusieurs fois***
   - **3.1.** Le message est lu sur la queue de sortie par différents composants et on se retrouve dans une situation de re-jeux.

4. ***Messages corrompus***
   - **4.1.** Une attaque « Man in the middle » et les messages ne sont plus intègres entre les deux queues.

5. ***Détournement des messages volontairement ou involontairement***
   - **5.1.** L'administrateur fait une erreur de configuration et les messages n'arrivent pas à l'endroit prévu.
   - **5.2.** Un hacker met en place un composant entre les deux queues et détourne le trafic.

6. ***Mauvaise gestion des certificats***
   - **6.1.** Le renouvellement des certificats n'est pas géré correctement et le service est indisponible.

7. ***Injection de messages non autorisés***
   - **7.1.** Une personne malveillante pourrait insérer des messages sur la queue de destination et donc by-passer tous les tests qui auraient été réalisés au préalable.

8. ***Plus d'espace***
   - **8.1.** Il n'y a plus d'espace au niveau de la source ou de la destination, ce qui bloque le système.
   - **8.2.** La consommation des messages ne suit pas leur arrivée, ce qui entraine un grossissement infini de la queue qui entrainera à son tour un manque de ressources.

E. Evaluation des scénarios

Tableau 25 : Scénarios message mover

Scénario de menaces	Source de menaces	Vraisemblance
1.1	- Infrastructure	Significative
1.2	- Infrastructure	Significative
1.3	- Administrateur	Minime
2.1	- Infrastructure	Minime
3.1	- Hacker	Significative
4.1	- Hacker	Significative
5.1	- Administrateur	Significative
5.2	- Hacker	Minime
6.1	- Administrateur	Significative
7.1	- Hacker	Significative
8.1	- Infrastructure	Significative
8.2	- Développeur - Administrateur	Significative

F. Analyse des risques

Tableau 26 : Risques message mover

Gravité	4. Critique	5.1	3.1 ; 4.1 ; 5.1 ; 6.1 ; 7.1 ; 8.1 ; 8.2		
	3. Importante	1.3 ; 2.1	1.1 ; 1.2		
	2. Limitée				
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.7 Processing Component

A. Présentation

L'idée principale du processing component est d'extraire la partie traitement en un composant à part entière. Ces sous-composants de traitement sont donc indépendants les uns par rapport aux autres. Chaque composant est connecté aux autres à l'aide de queues. Ces composants sont donc indépendants. On peut le voir aussi comme une méthode Java qui est caractérisée par des préconditions et des post-conditions. (Voir aussi le style architectural dit « micro service »).

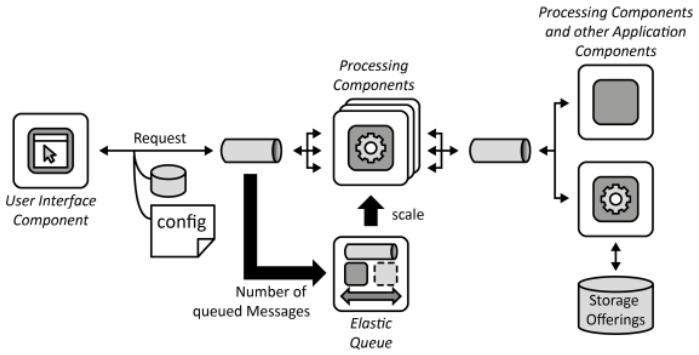

Figure 11 : Processing component (Fehling, 2014, p. 181)

B. Evènements redoutés applicables

1. Perte de messages (expérimentation).
2. Injection de codes [WASC, 2010].
3. Injection de structures de données mal structurées [WASC, 2010].
4. Accès à ces composants par des personnes non autorisées [CSA, 2013].
5. Création automatique pouvant couter chers, surtout si multiplication des composants [CSA, 2013].
6. Utilité d'un composant peut être détournée [WASC, 2010].
7. Queue indisponible (expérimentation).

C. Evaluation des évènements redoutés

Tableau 27 : Evènements redoutés processing component

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	Intègre	- Infrastructure - Administrateur	- Données perdues	Critique
2	Détectable	- Hacker	- Détournement du composant - Récupération de données	Limité
3	Détectable	- Hacker	- Transactions bloquées -Perte de données client	Limité
4	Limité	- Hacker	- Détournement de composants - Accès aux bases de données cachées derrière le composant	Critique
5	Financier	-Utilisateur (charge) - Hacker	- Surfacturation	Important
6	Intègre Financier Privé	- Hacker	- Détournement du composant - Fuite de données - Consommation des ressources	Limité
7	Moins 1h	- Infrastructure - Administrateur	- Indisponibilité du service	Critique

D. Scénarios par évènement

1. Perte de messages

1.1. Le disque dur de destination est plein, il n'est plus possible de persister le message au sein de queues.

1.2. Dans le cadre d'une communication UDP, la perte de paquets réseaux est possible.

1.3. Mauvaise configuration des queues qui entraine une perte de message.

2. Injection de codes

2.1. La requête envoyée est hostile. Une injection de code est faite. Dans le cas où le processing component exécute sans validation des données, le comportement du composant n'est pas garanti.

3. ***Injection de structures de données mal structurées***

3.1. Une personne mal intentionnée soumet des requêtes manipulées.

4. ***Accès aux composants par des personnes non autorisées***

4.1. Une personne, déjà au sein du réseau, pourrait utiliser les composants de façon indépendante et accéder à des données non autorisées.

5. **Création automatique pouvant couter cher, surtout si multiplication des composants**

5.1. Le nombre d'utilisateur s'accroît significativement et donc la facturation aussi.

5.2. L'infrastructure subit une attaque DDOS et la facturation en est impactée même si le service est toujours disponible.

6. ***Création automatique pouvant être couteuse, surtout si multiplication des composants***

6.1. Un utilisateur malveillant fait une utilisation anormale d'un composant. Il s'agit d'attaques en parallèle avec d'autres méthodes. Les effets de bord sont donc multiples et on pourrait avoir une fuite de données, un blocage de l'infrastructure ou encore donner l'accès à une partie de l'application à une personne non autorisée.

7. **Voir les scénarios « message mover »**.

E. Evaluation des scénarios

Tableau 28 : Scénarios processing components

Scénario de menaces	**Source de menaces**	**Vraisemblance**
1.1	- Infrastructure	Significative
1.2	- Infrastructure	Significative
1.3	- Administrateur	Minime
2.1	- Hacker - Utilisateur	Forte
3.1	- Hacker	Forte
4.1	- Menace interne	Significative
5.1	- Utilisateur	Significative
5.2	- Hacker	Forte
6.1	- Hacker	Minime
7	Voir message mover	

F. Analyse des risques

Tableau 29 : Risques processing components

Gravité	4. Critique	1.3	1.1 ; 1.2 ; 4.1		
	3. Importante		5.1 ; 5.2		
	2. Limitée	6.1		2.1 ; 3.1	
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.8 Data Access Component

A. Présentation

Ce pattern a pour but de mettre en place une interface entre la base de données et le code applicatif dans le but de dissimuler la complexité de l'accès aux données. Il permet ainsi une plus grande indépendance de l'application par rapport à l'offre de storage utilisée. L'application sera donc plus simple à maintenir et sera à même de faire face à des systèmes complexes de type HSM (Hierarchical Storage Management) ou multi-instances.

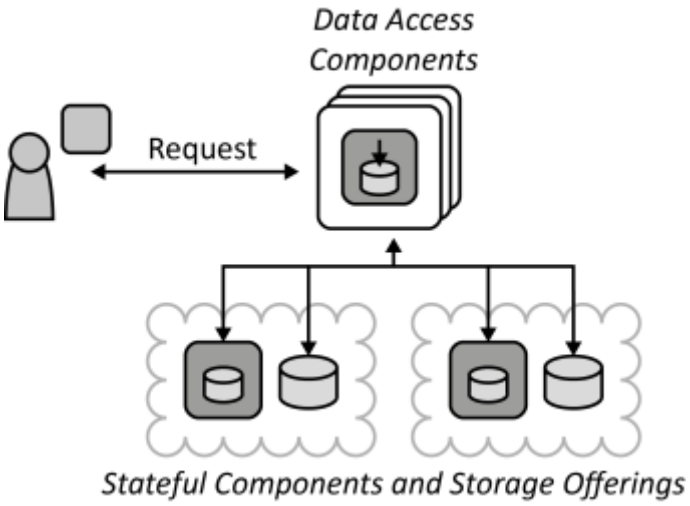

Figure 12 : Data access components integrating stateful components and storage offering residing in two clouds (Fehling, 2014, p. 189)

B. Evènements redoutés applicables

1. Transaction mal gérée (expérimentation).
2. Limites de l'infrastructure (expérimentation).
3. Accès aux composants par des personnes non autorisées [CSA, 2013].
4. Requêtes mal formées via une méthode d'injection [WASC, 2010].

C. Evaluation des évènements redoutés

Tableau 30 : Evènements redoutés data access components

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	- Intègre - Moins 1h	- Développeur - Administrateur	- Transaction bloquée - Intégrité entre les différentes instances de stockage	Important
2	- Moins 1h	- Infrastructure - Utilisateurs	- Réponse aux requêtes trop lentes	Limite
3	- Privé	- Hacker - Administrateur	- Fuite de données	Critique
4	- Intègre - Privé - Moins 1h	- Hacker	- Blocage des transactions - Intégrité des données - Indisponibilité - Fuite de données	Critique

D. Scénarios par évènement

1. Transaction mal gérée

1.1. Dans le cas de problèmes d'accès concurrents, si le développeur ne tient pas compte et ne crée pas une transaction distribuée entre les différentes instances de bases de données, on peut se retrouver dans une situation où la transaction est bloquée ou inconsistante.

1.2. Dans le cas où la transaction manager est mal configurée par l'administrateur, il peut y avoir soit une transaction bloquée, soit une inconsistance entre les instances des bases de données.

2. Limites de l'infrastructure

2.1. Le nombre de requêtes est supérieur à ce que l'infrastructure permet de traiter, ce qui entraine une instabilité voire une indisponibilité. Cela peut être la conséquence de différentes attaques en amont.

3. Accès aux composants par des personnes non autorisées

3.1. L'accès aux données par un administrateur à des fins malveillantes.

3.2. Un hacker déjà dans le système accède aux bases de données.

3.3. Un hacker utilise une attaque « man in the middle » pour accéder aux données à la place d'un autre utilisateur.

4. Requêtes mal formées via une méthode d'injection

4.1. Un hacker utilise les différentes techniques d'injection (sql, SOAP, xml) dans le but de modifier le résultat ou de modifier les données à persister.

E. Evaluation des scénarios

Tableau 31 : Scénarios data access components

Scénario de menaces	Source de menaces	Vraisemblance
1.1	- Développeur	Minime
1.2	- Administrateur	Minime
2.1	- Infrastructure	Significative
3.1	- Administrateur	Minime
3.2	- Hacker	Minime
3.3	- Hacker	Significative
4.1	- Hacker	Significative

F. Analyse des risques

Tableau 32 : Risques data access components

Gravité	4. Critique	3.1 ; 3.2	3.3 ; 4.1		
	3. Importante	1.1 ; 1.2			
	2. Limitée		2.1		
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.9 Two-Tier Cloud Application

A. Présentation

Dans cette configuration d'un « two-tier cloud application », les parties présentation et métier sont liées comme un seul composant. La partie persistance, quant à elle, est distinguée. Le fait de combiner la vue et le métier permet de simplifier la gestion d'auto provisioning au niveau de la partie logicielle. Cette solution est donc complètement incompatible avec le pattern state full, car dans ce contexte, chaque instance de l'application est totalement indépendante par rapport aux autres. Les données séparées permettent une plus grande souplesse dans la gestion de celles-ci.

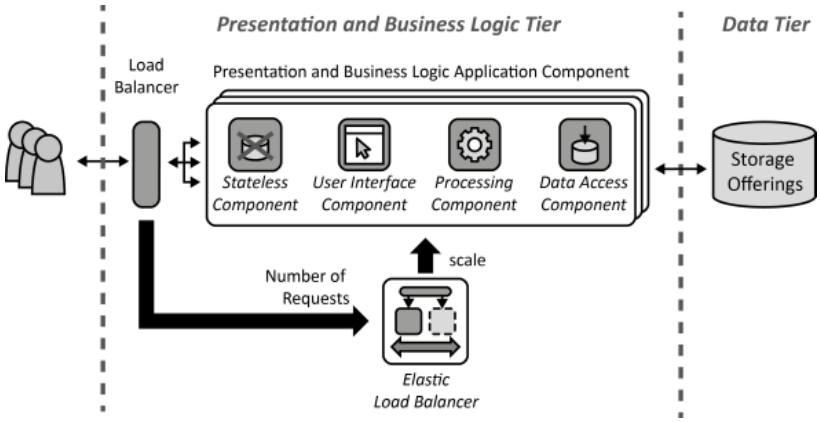

Figure 13 : Two-tier architecture (Fehling, 2014, p. 291)

B. Evènements redoutés applicables

1. « Load balancing failure » (expérimentation).
2. Explosion de la facturation en cas d'attaque ou d'utilisation trop grande par rapport au budget de l'entreprise [CSA, 2013].
3. Faille de connexions entre les instances et les bases de données (expérimentation).

C. Evaluation des évènements redoutés

Tableau 33 : Evènements redoutés two-tier architecture

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	Moins 1h	- Administrateur	- Indisponibilité du service	Critique
2	Financier	- Hacker - Utilisateur	- Facturation non maitrisée	Important
3	Moins 1h	- Infrastructure - Administrateur	- Persistance impossible	Limité

D. Scénarios par évènement

1. ***« Load balancing failure »***
   1.1. La configuration du load-balancing ne prendrait pas bien en compte l'augmentation du nombre de requêtes. Le mécanisme d'auto scalling ne fonctionnerait pas et une indisponibilité peut donc en résulter.

2. ***Explosion de la facturation en cas d'attaque ou d'utilisation trop grande par rapport au budget de l'entreprise***
   2.1. Dans l'hypothèse où le système subit une attaque de type DDOS, de par l'élasticité de la solution proposée, le nombre d'instances pourrait augmenter dans le but de pouvoir répondre aux requêtes. Cela impacterait la facturation de la solution, voire une indisponibilité si certaines règles ont été configurées.

3. ***Faille de connexions entre les instances et les bases de données***
   3.1. La connexion réseau est interrompue.
   3.2. Un administrateur change le mot de passe de la base de données.

E. Evaluation des scénarios

Tableau 34 : Scénarios two-tier architecture

Scénario de menaces	Source de menaces	Vraisemblance
1.1	- Administrateur - Infrastructure	Significative
2.1	- Hacker	Maximale
3.1	- Infrastructure	Significative
3.2	- Administrateur	Significative

F. Analyse des risques

Tableau 35 : Risques two-tier architecture

Gravité	4. Critique		1.1		
	3. Importante				2.1
	2. Limitée		3.1 ; 3.2		
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.10 Three-Tier Cloud Application

A. Présentation

Dans le cas du « three-tier cloud application », les trois couches, à savoir la présentation, le traitement et l'accès aux données sont séparées les unes des autres, et ce dans le but de profiter au maximum des propriétés d'élasticité du cloud. En effet, certaines applications ne vont pas générer la même charge sur chaque couche. Il se peut que certaines applications aient des composants de traitement beaucoup plus gourmands que la couche de présentation. Dans cette solution, la communication entre les différentes couches se fait à l'aide de messages. Elle ne sera donc pas adaptée au composant statefull. Dans cette présentation, il est à noter que différentes présentations peuvent accéder à la même couche logique, tout comme différents types de couches logiques peuvent accéder à la même couche de données. C'est le modèle idéal dans les systèmes hautement distribués que sont les applications modernes car il favorise la réutilisation de composant.

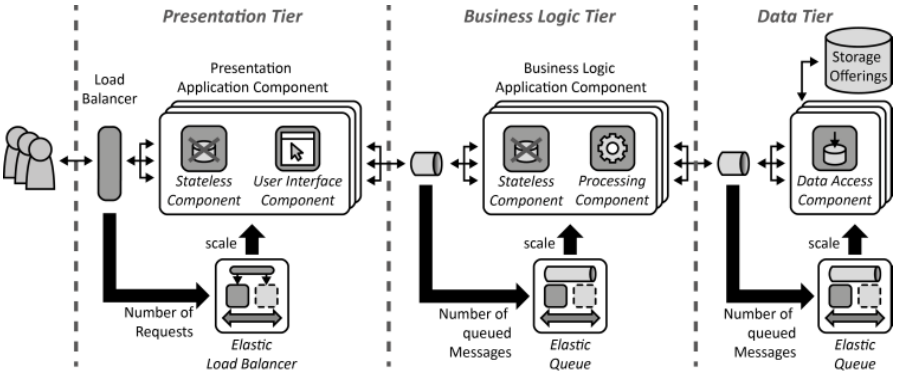

Figure 14 : Three-tier architecture (Fehling, 2014, p. 295)

B. Evènements redoutés applicables

1. Problème de consistance entre les différentes instances de la base de données (expérimentation).
2. Système asynchrone de par les queues entre chaque couche (voir les risques liés au pattern message mover).
3. Accès à une des couches par un tiers. [CSA, 2013]
4. Problème au load balancer (voir les risque liés au pattern Elastic Load Balancer).

C. Evaluation des évènements redoutés

Tableau 36 : Evènements redoutés three-tier architecture

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	Intègre	- Développeur - Administrateur	- Perte d'intégrité entre les différentes instances de la base de données	Critique
2	Intègre	- Administrateur	- Perte de messages - Arrêt de traitement des requêtes	Critique
3	Privé	- Hacker	- Détournement de données	Critique
4	Moins 1h	- Administrateur - Infrastructure	- Service indisponible	Important

D. Scénarios par évènement

1. ***Problème de consistance entre les différentes instances de la base de données***
   - **1.1.** De par le fait que la couche data est évolutive, on peut imaginer que les données ne sont pas intègres entre les instances. Si l'application est mal écrite, il y aura une mauvaise gestion des transactions.
   - **1.2.** La configuration des transactions entre les instances de la base de données étant mal faite, il peut en résulter un manque d'intégrité entre les instances.

2. ***Système asynchrone de par les queues entre chaque couche (Voir les risques liés au pattern message mover)***
   - **2.1.** Voir les scénarios liés au « message mover pattern ».

3. ***Accès à une des couches par un tiers***
   - **3.1.** Une personne mal intentionnée qui aurait accès à l'intérieur du réseau pourrait se connecter sur les différentes couches sans passer par la première afin de détourner des informations.

4. ***Queue indisponible***
   - **4.1.** Voir les scénarios liés au « elastic load balancer pattern ».

E. Evaluation des scénarios

Tableau 37 : Scénarios three-tier architecture

Scénario de menaces	Source de menaces	Vraisemblance
1.1	- Infrastructure - Administrateur - Développeur	Significative
1.2	- Développeur	Minime
2.1	Voir message mover	
3.1	- Hacker - Administrateur	Minime
4.1	Voir elastic load balancer	

F. Analyse des risques

Tableau 38 : Risques three-tier architecture

Gravité	4. Critique	1.2 ; 3.1	1.1 ; 2.1		
	3. Importante				
	2. Limitée				
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

5.2.11 Hybrid Development Environment

A. Présentation

L'environnement de développement étant généralement une copie de l'environnement de production avec toutes ses dépendances, tous ses composants sont souvent inutilisés à part en phase de tests. Nous faisons appel à l'une des propriétés du cloud qui permet une élasticité au niveau des ressources, ainsi qu'à l'auto provisioning. Cette solution permet dans le cas de l'utilisation de la même image de garantir que tous les tests réalisés dans l'environnement de développement seront représentatifs de ce qui se fait en production. Il sera aussi plus aisé d'exécuter des tests sur différents environnements.

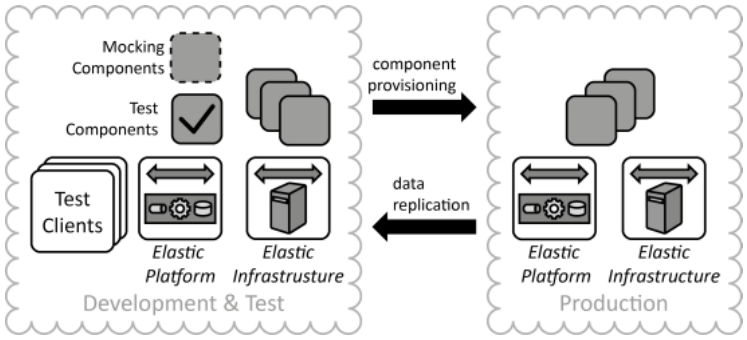

Figure 15 : Hybrid development environment and its integration with a production environment (Fehling, 2014, p. 327)

B. Evènements redoutés applicables

1. Les données de production sont accessibles à tous les utilisateurs du cloud de développement (expérimentation).
2. Les données peuvent être détournées [CSA, 2013].
3. Les mots de passe sont plus faibles sur le cloud de développement. Leur utilisation quotidienne par les utilisateurs risque donc de se retrouver sur des post-it ou dans des mails (expérimentation).
4. Détournement des mocks [WASC, 2010].

C. Evaluation des évènements redoutés

Tableau 39 : Evènements redoutés hybrid development environment

Évènement redouté	Besoin de sécurité	Source de menaces	Impact	Niveau gravité
1	Privé	- Développeur	- Accès aux personnes non autorisées aux données client	Important
2	Privé	- Développeur - Administrateur - Service provider	- Perte de contrôle sur les données	Critique
3	Privé	- Développeur - Hacker	- Accès simple au système miroir de la production - Accès au code	Important
4	Intégrité	- Hacker	- Utilisation des mocks en production - Détournement du code	Important

D. Scénarios par évènement

1. ***Les données de production sont accessibles à tous les utilisateurs du cloud de développement***
   - **1.1.** Imaginons qu'il n'y ait aucun filtre, alors les données de production seront accessibles à un plus grand nombre de personnes. Le mot de passe des environnements de test sont rarement très sûrs.

2. ***Les données peuvent être détournées***
   - **2.1.** On peut facilement imaginer une perte des données due à l'accès simple aux données de production.

3. ***Les mots de passe sont plus faibles sur le cloud de développement. Leur utilisation quotidienne par les utilisateurs risque donc de se retrouver sur des post-it ou dans des mails.***
   - **3.1.** Le fait que les mots de passe des environnements de développement soient faibles entraine la possibilité de remplacer le code de l'application qui sera par la suite déployé en production.

4. Détournement des mocks

4.1. Imaginons un hacker qui aurait accès aux environnements de développement et qui détournerait les mocks pour manipuler la production.

E. Evaluation des scénarios

Tableau 40 : Scénarios hybrid development environment

Scénario de menaces	Source de menaces	Vraisemblance
1.1	- Développeur - Utilisateur interne	Minime
2.1	- Développeur	Minime
3.1	- Utilisateur interne - Hacker	Minime
4.1	- Hacker	Minime

F. Analyse des risques

Tableau 41 : Risques hybrid development environment

Gravité	4. Critique	2.1			
	3. Importante	1.1 ; 3.1 ; 4.1			
	2. Limitée				
	1. Négligeable				
		1. Minime	2. Significative	3. Forte	4. Maximale
	Vraisemblance				

6 Discussion

Cette recherche a tenté de mettre en pratique une méthode d'analyse de risques dans un contexte particulier, à savoir les « Cloud Design Patterns ». Nous avons utilisé la méthode d'analyse EBIOS qui semblait être la plus appropriée à ce contexte. En effet, la démarche présentée a permis de structurer l'analyse tandis que les définitions proposées ont donné un cadre pour l'évaluation de chaque situation. Nous nous trouvons ainsi en présence d'une base de connaissance des risques liés à des patterns de développement.

Les patterns étant à l'origine de toutes les nouvelles applications, il nous semblait pertinent d'analyser les risques encourus par les patterns car ceux-ci représentent les fondations d'une application. La base de connaissance des risques en lien avec chaque pattern, construite au cours de l'analyse, sera utile à toutes les personnes qui utilisent des patterns au cours de l'élaboration d'une nouvelle application.

En effet, lors de la première phase de développement consistant à choisir entre les différents patterns disponibles, l'architecte pourra se référencer aux risques liés à chacun d'entre eux afin d'effectuer son choix. Par exemple, dans le cadre d'une application web, est-il préférable d'utiliser un statefull component ou bien un stateless component ? Grâce à l'analyse réalisée, l'architecte pourra évaluer les risques encourus en fonction du pattern qu'il compte retenir pour créer son système d'information. Si l'architecte doit développer une application sur le cloud qui doit pourvoir supporter une élasticité du nombre d'instances applicatives, l'analyse mettra en avant l'utilisation du « stateless component ».

Une fois que tous les patterns composant le nouveau système ont été sélectionnés, il faut alors prendre en compte la liste des risques encourus par l'ensemble de l'architecture. Sur base de ce listing, des contre-mesures doivent être mises en place afin de réduire le nombre des risques résiduels à un niveau acceptable pour la mise en production.

Pendant toute la durée de la maintenance, la liste des risques résiduels doit être suivie car de nouveaux risques pourraient apparaitre au cours de l'exploitation de l'application.

Sur la base de l'analyse réalisée dans cette recherche, nous voyons donc qu'il est possible d'élaborer un listing des risques de sécurité pour chaque application et de maintenir celui-ci à jour durant toute la durée de vie de l'application.

Néanmoins, cette analyse présente quelques limites. En effet, elle n'est pas exhaustive et elle doit être enrichie. En effet, elle ne couvre pas tous les cas de figure que pourrait présenter chaque pattern. Dans le but de couvrir au mieux tous les risques encourus, différents profils d'experts (des architectes, des administrateurs, des experts en sécurité, des experts réseau) pourraient participer à cette analyse et ainsi la compléter avec leur spécificité.

Quant à la base des patterns couverts, elle pourrait également être étendue en invitant d'autres sociétés qui utilisent des patterns plus spécifiques. Effectivement, agrandir cette base de connaissance aiderait les architectes et les développeurs à mieux prévoir les contre-mesures afin de limiter au maximum les risques de sécurité.

Cette base de connaissance pourrait appartenir à la structure du pattern. Comme un pattern fait référence à d'autres patterns en lien avec lui-même, il pourrait également contenir la liste de risques liés à son utilisation. Cela permettrait une mise en commun des risques liés à chaque pattern.

De plus, le caractère cyclique inhérent à l'analyse des risques empêche de terminer cette étude. En effet, au vu de la rapidité de l'évolution de la technologie et des recherches, il est difficile de prévoir tous les risques liés à la sécurité. De fait, nous sommes en présence d'un processus en constante évolution. D'ailleurs au moment de la rédaction de cette discussion, une publication de la KUL [Vanhoef, 2016] met en avant une nouvelle faille « HEIST » au niveau du protocole « HTTPS ». Dès lors, si on veut aider les architectes et les administrateurs à développer des systèmes d'information fiables et sécurisés, la liste des risques devra être constamment actualisée et des contre-mesures devront sans cesse être apportées.

Cette analyse a mis en avant les risques présents pour chaque pattern. Dès lors, comme nous l'avons décrit au début de la discussion, les responsables des applications devraient traiter ces risques en partant de la mise en place de contre-mesures. Or, dans cette analyse, les contre-mesures n'ont pas été listées. Il serait donc utile de les répertorier afin que les concepteurs puissent les mettre en place de manière systématique lors de l'élaboration d'un nouveau système.

Une autre piste serait d'automatiser l'analyse de l'architecture dans le but de garder à jour la liste des risques associés à chaque application. Comme nous l'avons vu précédemment, cette liste des risques par pattern évolue en fonction de l'avancée des recherches. Cette automatisation permettrait un meilleur suivi des risques. Elle pourrait, entre autres, prendre la même forme que celle utilisée pour l'analyse du code déjà présente actuellement à savoir la duplication du code, le niveau de documentation, la gestion des dépendances, etc …

7 Conclusion

A l'heure où de plus en plus d'entreprises ont recours au cloud, on comprend tout l'intérêt de s'intéresser aux risques de sécurité qu'il engendre.

Cette recherche a tenté de mettre en pratique une méthode d'analyse dans un contexte particulier, à savoir les « Cloud Design Patterns ». Nous avons utilisé la méthode d'analyse EBIOS qui semblait être la plus appropriée à ce contexte. En effet, la démarche présentée a permis de structurer l'analyse et les définitions proposées ont donné un cadre pour l'évaluation de chaque situation.

L'originalité du travail réside dans le fait que nous avons essayé de mettre en évidence les risques liés à la sécurité à partir de l'analyse de patterns architecturaux alors que les différentes études existantes s'attachent à l'infrastructure du cloud.

Les patterns retenus sont issus du livre « Cloud Computing Pattern » [Fehling, 2014] et ont été expérimentés au sein de deux entreprises. Il s'agit des patterns suivants : « two-tier cloud application », « three-tier cloud application » et « hybrid developement environment ». Vu que chacun de ces trois patterns est également composé de différents patterns appelés « components patterns », au total, nous avons analysé dix patterns.

Avoir travaillé en entreprise sur ces différents patterns a permis d'insérer des évènements redoutés qui ne semblent pas avoir été repris dans la littérature scientifique. Chaque évènement redouté a été attaché à un ou plusieurs acteurs permettant d'élaborer un certain nombre de scénarios en fonction de l'évènement mais aussi de la source de menace.

Lors de l'évaluation, chaque pattern a obtenu une cotation liée au risque pour chacun des scénarios développés lors de l'analyse. Nous avons également mis en avant qu'un simple component pattern peut encourir plusieurs risques, et ce avec des niveaux différents. Lors de l'utilisation d'un tel pattern au sein d'un pattern architectural, ce dernier hérite des risques du premier, ce qui rend notre analyse pertinente. Nous avons donc commencé par la plus petite partie d'un système d'information, à savoir un component pattern, avant de remonter vers un système plus complexe, le pattern architectural. On peut en déduire qu'il faut concevoir les contre-mesures de sécurité en même temps que le système d'information, et ce en fonction des choix architecturaux qui auront été faits. De plus, les contre-mesures devront être prises à chaque niveau afin d'éviter l'héritage de ces risques au niveau des patterns architecturaux. En effet, après l'analyse de chaque component pattern, des contre-

mesures doivent être prises pour traiter les risques, et ce afin d'éviter que ceux-ci ne se propagent au niveau des patterns architecturaux.

Dès lors, si on veut aider les architectes et les administrateurs à développer des systèmes d'information fiables et sécurisés, la liste des risques devra être constamment actualisée et des contre-mesures devront sans cesse être apportées.

8 Bibliographie

- [WASC, 2010] “WASC Threat Classification Version 2.0.” Web Application Sécurity Consorcum (2010).
- [CSA, 2013] “The Notorious Nine cloud Computing Top Threats in 2013.” Cloud Security Alliance (2013).
- [Rappa, 2004] M. A. Rappa “The utility business model and the future of computing services.” IBM Systems Journal VOL 43, No 1 (2004):32-42.
- [Perez-Boteron, 2013] Diego Perez-Botero, Jakub Szefer and Ruby B.Lee. “Characterizing Hypervisor Vulnerabilities in Cloud Computing Servers.” Proceedings of the Workshop on Security in Cloud Computing (SCC), May 2013.
- [Nguyen, 2014] Minh-Duong Nguyen, Ngoc-Tu Chau, Seungwook Jung, and Souhwan Jung “A Demonstration of Malicious Insider Attacks inside Cloud Iaas Vendor” International Journal of Information and Eduction Technology, VOL 4, No 6 (December 2014): 483-486
- [SGDN / ANSSI / ACE / BAC, 2011] “Etude de cas : Sécurité d’un service du Cloud.” Secrétariat général de la défense et de la sécurité nationale / Agence nationale de la sécurité des systèmes d’information / Sous-direction assistance, conseil et expertise / Bureau assistance et conseil. (2011)
- [SGDN / ANSSI / ACE / BAC, 2010] “Expression des Besoins et Identification des Objectifs de Sécurité. ” Secrétariat général de la défense et de la sécurité nationale / Agence nationale de la sécurité des systèmes d’information / Sous-direction assistance, conseil et expertise / Bureau assistance et conseil. (2010)
- [NIST, 2011] The NIST Definition of Cloud Computing, NIST, 2011.
- [Mayer, 2006] Nicolas Mayer, Jean-Philippe Humbert “La gestion des risques pour les systèmes d’information. ” MISC No 24 (2006)
- [Kholia, 2013] Dhiru Kholia, Przemyslaw Wegrzyn “Looking inside the (Drop) box. ” openwall (2013).
- [Izrailevsky, 2016] Yury Izrailevsky “Fin de la migration de Netflix vers le cloud. ” https://media.netflix.com/fr/company-blog/completing-the-netflix-cloud-migration (février 2016).
- [Homer, 2014] Alex Homer, John Sharp, Larry Brader, Masashi Narumoto, Trent Swanson “*Cloud Design Patterns.”* Microsoft. 978-1-62114-036-8
- [Ghaddar, 2012] Ali Ghaddar, Dalila Tamzalit, Ali Assaf “Gestion de la variabilité dans les applications SaaS multi-locataires.” (2012)
- [Collin, 2015] Jean noël Collin “Sécurité des systèmes informatique.” UNamur (2015) :16-44
- [CLUSIF, 2010] “MEHARI 2010 : Manuel de référence des Services de Sécurité.”, Clu de la sécurité de l’information français (2010).

- [Fehling, 2014] Christoph Fehling, frank Leymann, Ralph Retter, Walter Schupeck, Peter Arbitter “Cloud Computing Patterns.” Springer 978-3-7091-1567-1
- [SEI, 2007] Richard A. Caralli, James F. Stevens, Lisa R. Young, William R. Willson “Introducting OCTAVE Allegro: Improving the Information Security Risk Assessment Process” Software Engineering Institute (2007)
- [SAFECode, 2011] Mark Belk, Matt Coles, Cassio Goldschmidt, Michael Howard, Kyle Randolph, Mikko Saario, Reeny Sondhi, Izar Tarandach, Antti Vähä-Sipilä, Yonko Yonchev “ Fundamental Practices for Secure Software Development” Software Assurance Forum for Excellence in Code (2011)
- [Armbrust, 2009] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia “Above the Clouds: A Berkeley View of Cloud Computing” UC Berkeley Reliable Adaptive Distributed Systems Laboratory (2009).
- [ARBOR, 2015] ARBOR Network, “Worldwide Infrastructure Security Report” ARBOR Report (2015)
- [SEI, 2003] Christopher Alberts, Dorofee Audrey, James Stevens, Carol Woody “Introduction to the OCTAVE Approach” Software Engineering Institute (2003).
- [ANSI, 2014] “Comparatif des Méthodes d’audit et d’analyse des risques de sécurité des systèmes d’information.” Agence Nationale de la Sécurité Informatique tunisien (2014).
- [CERT, 2016] “Insider thread” definition http://www.cert.org/insider-threat/ (2016, 03 06)
- [Vanhoef, 2016] Mathy Vanhoef, Tom Van Goethem “HEIST: HTTP Encrypted Information can be Stolen through TCP-windows” iMinds-DistriNet (2016).